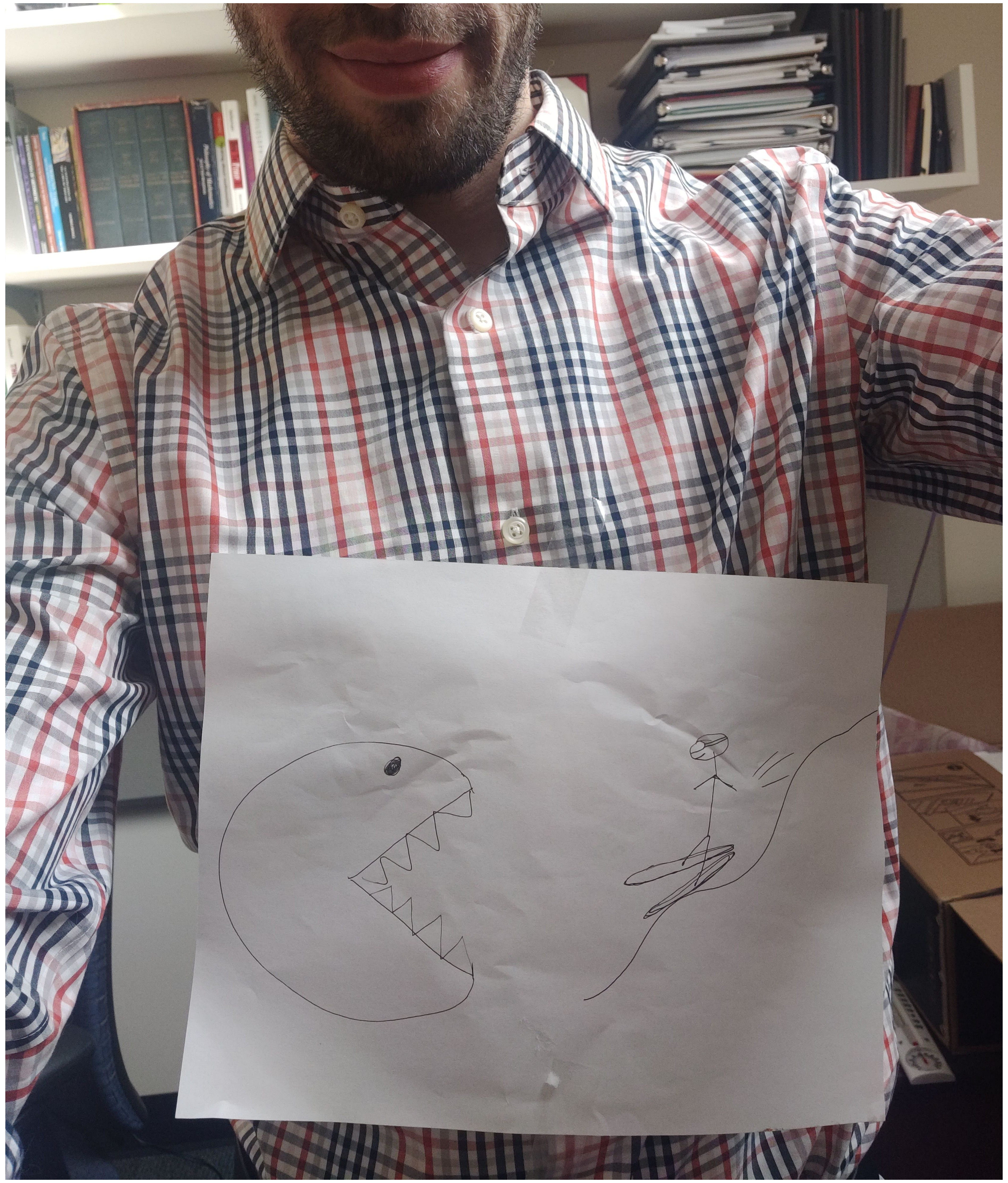
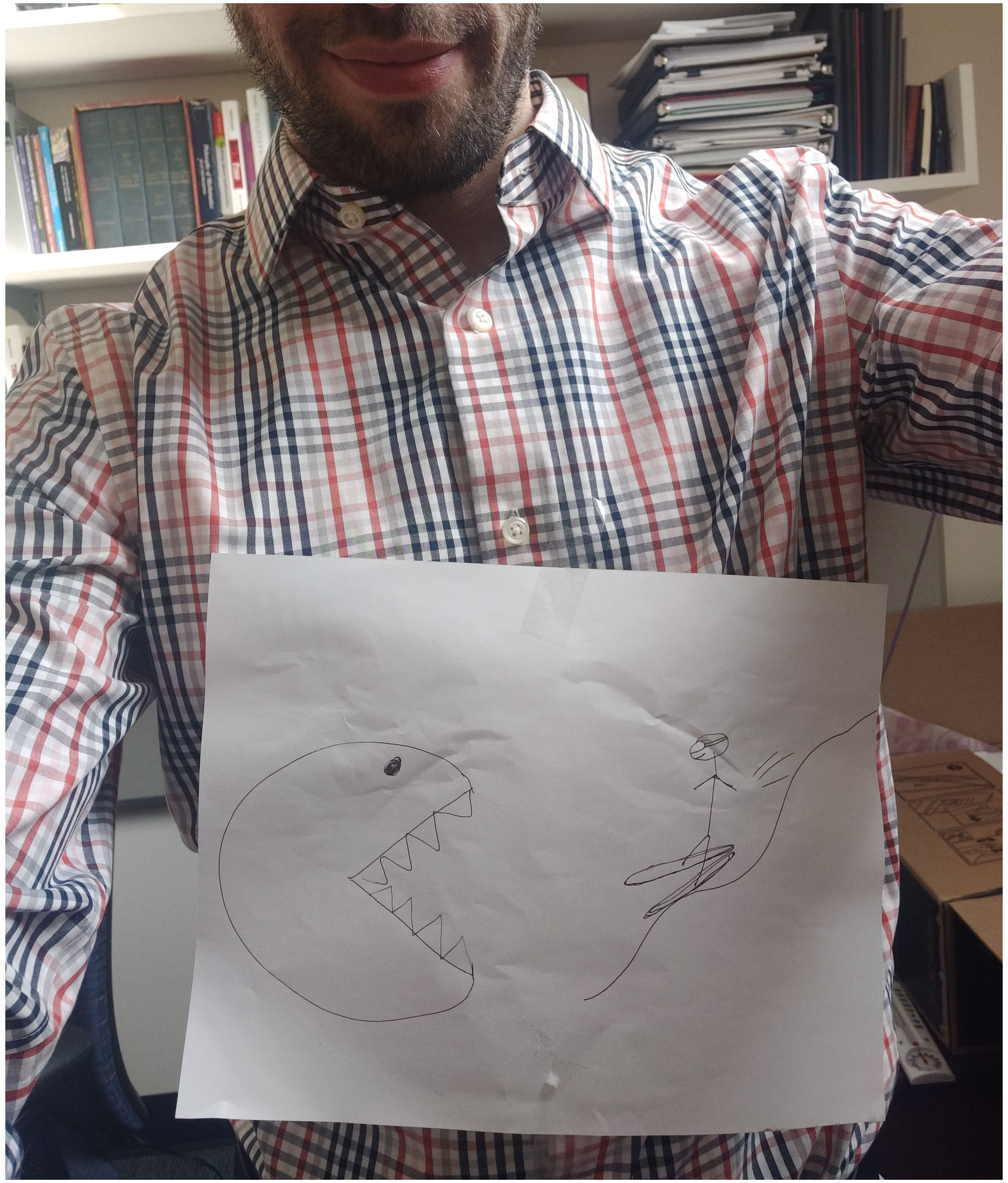
Feature-based Parsing
+
Computational Semantics

LING 571 — Deep Processing for NLP
October 27, 2021
Shane Steinert-Threlkeld

Happy (early) Halloween!



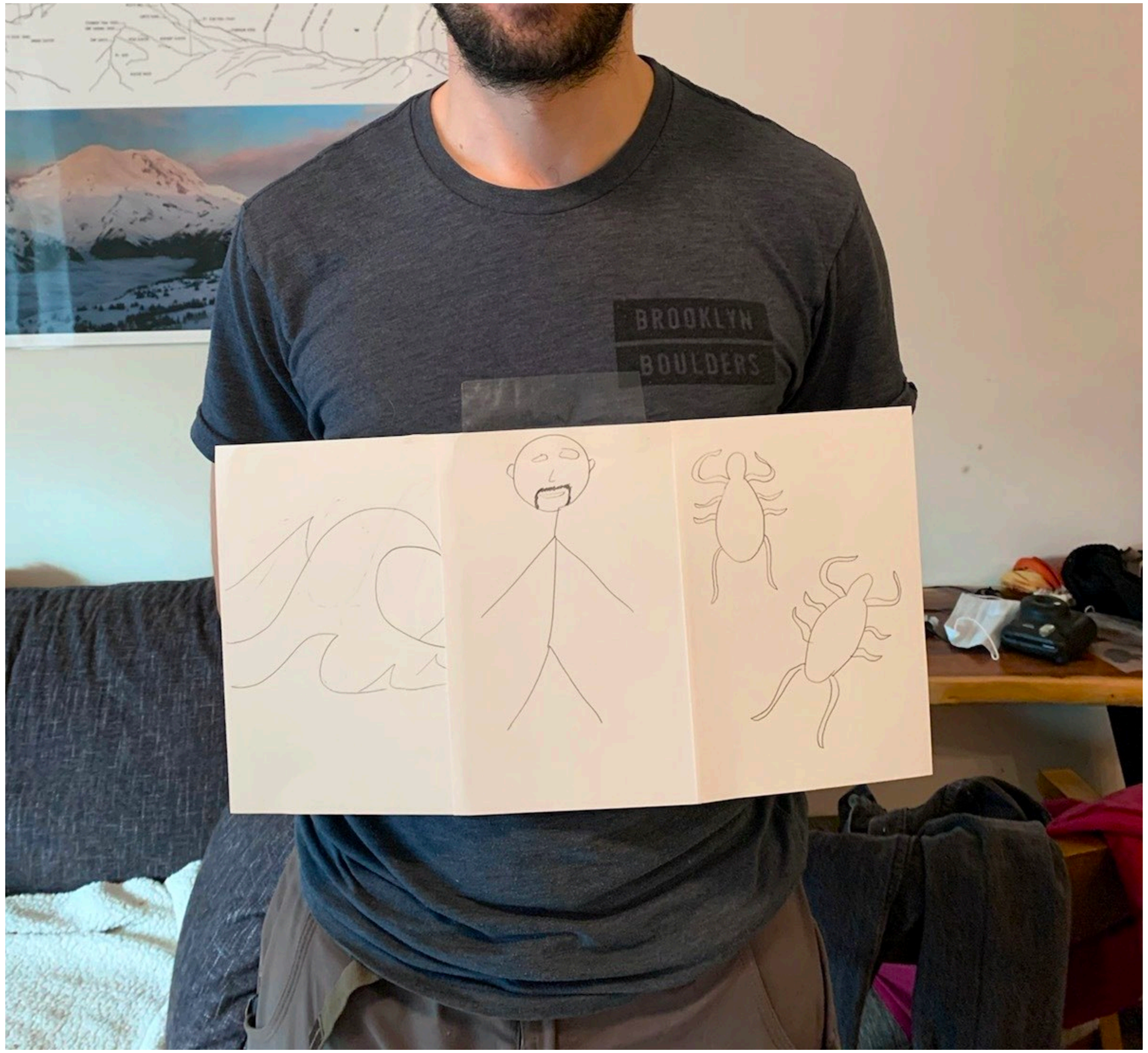
Happy (early) Halloween!



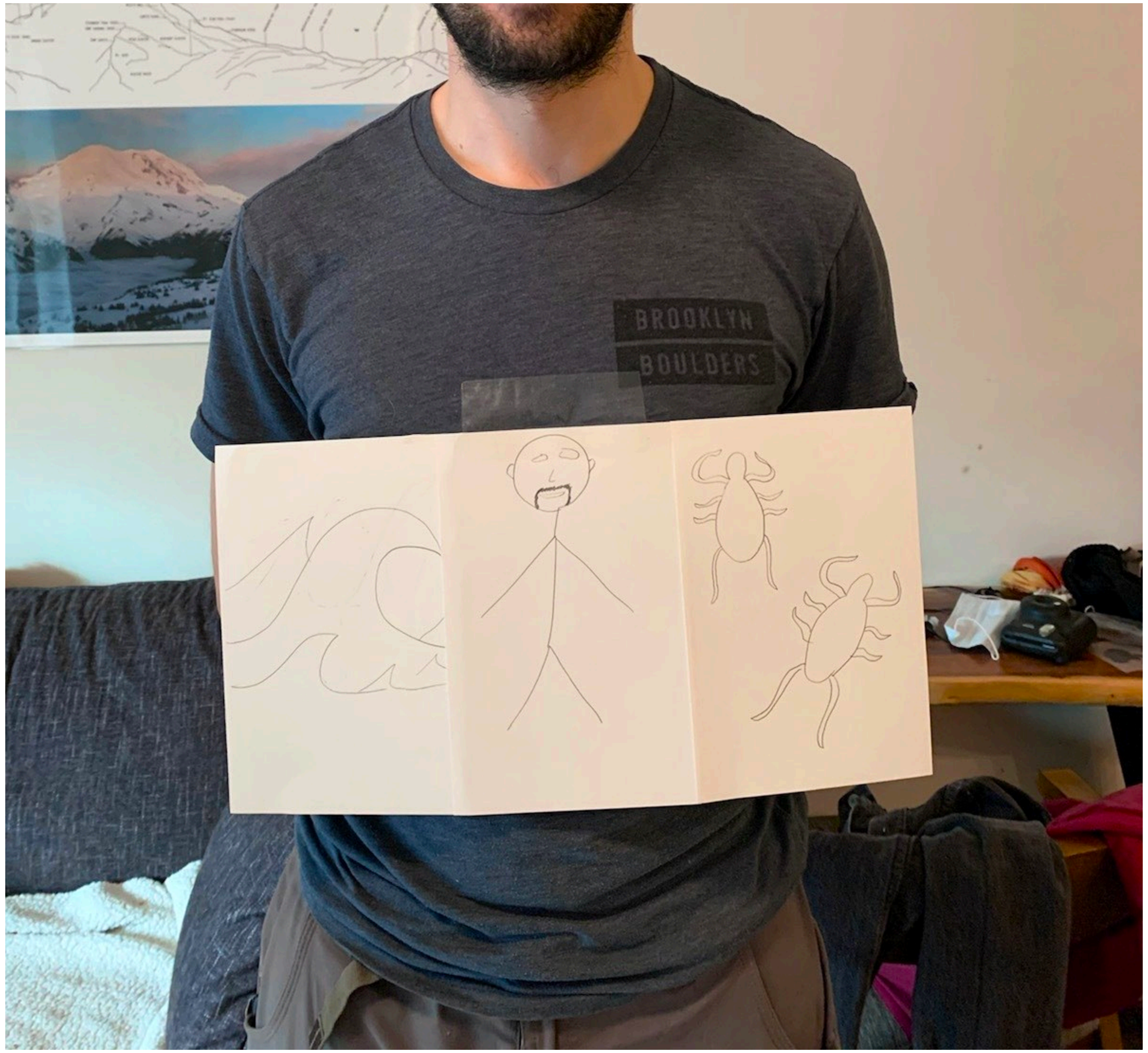
2019: Chomp + Ski = Chomsky

Punny Department

Happy (early) Halloween!

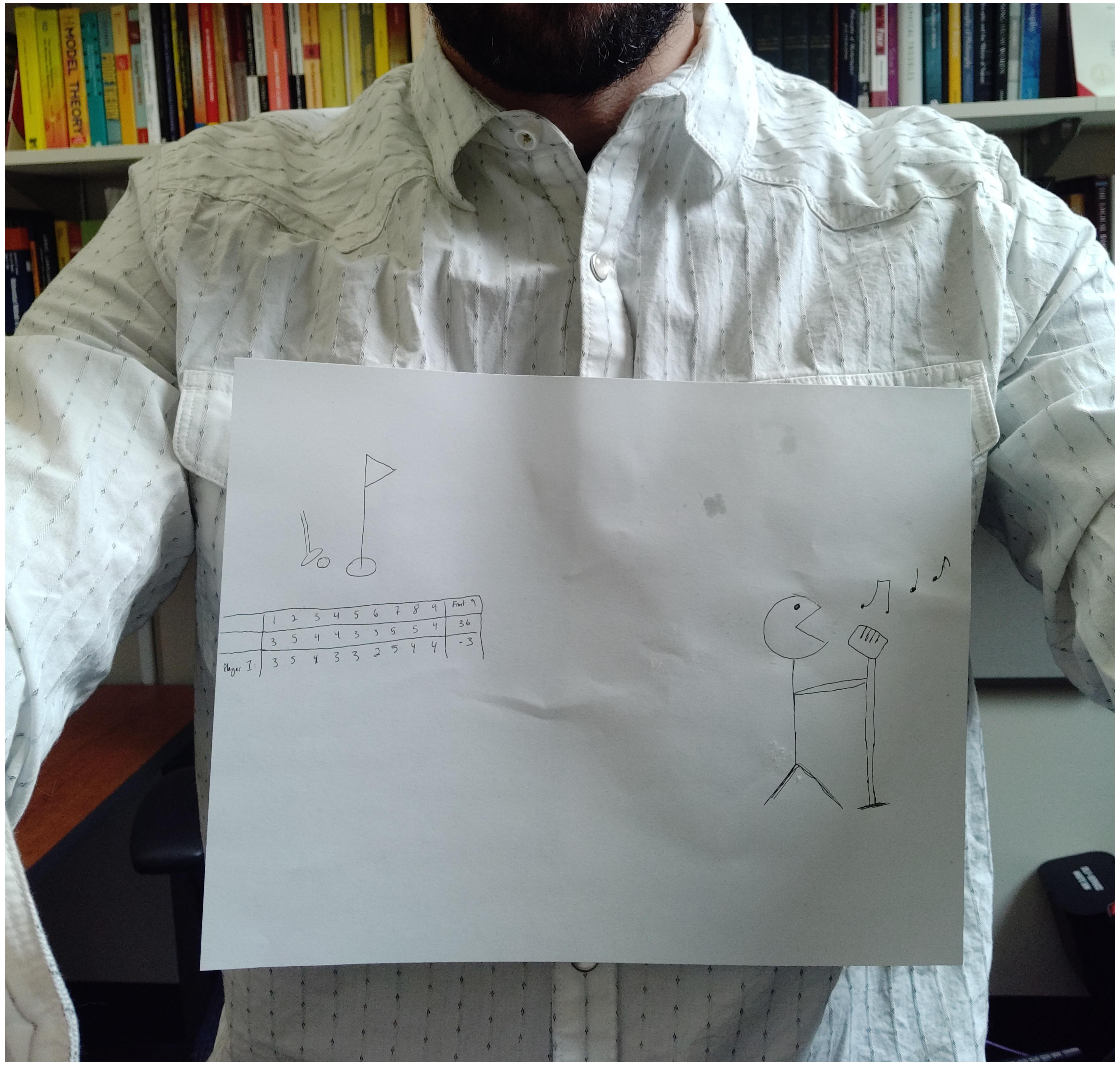


Happy (early) Halloween!

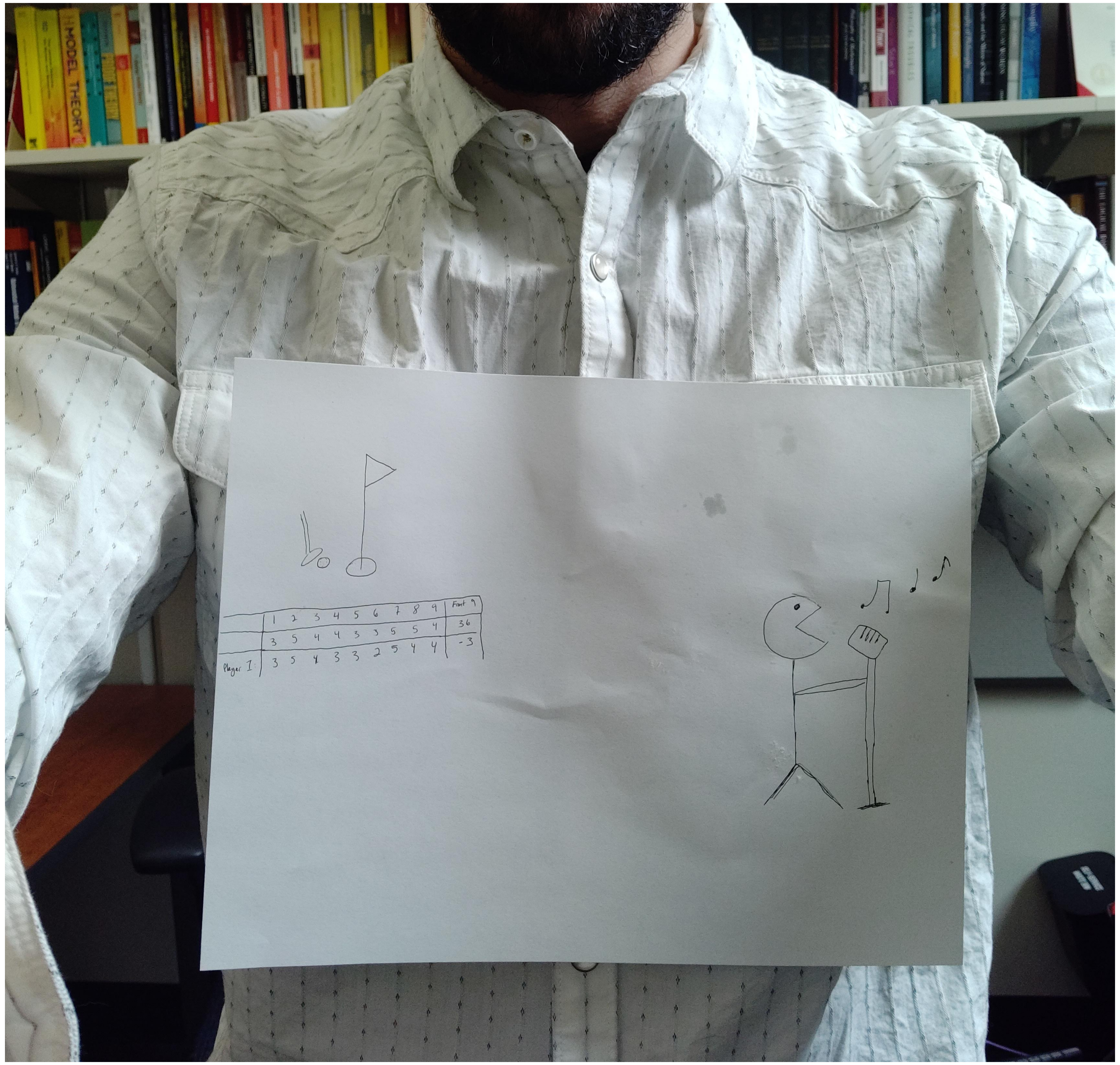


2020: Sea + Man + Ticks = Semantics

Happy (early) Halloween!



Happy (early) Halloween!



2021: ???

Guess the costume (one word)!

Roadmap

- Feature-based parsing
- Computational Semantics
  - Introduction
  - Semantics
  - Representing Meaning
    - First-Order Logic
    - Events
- HW#5
  - Feature grammars in NLTK
  - Practice with animacy

Computational Semantics

Dialogue System

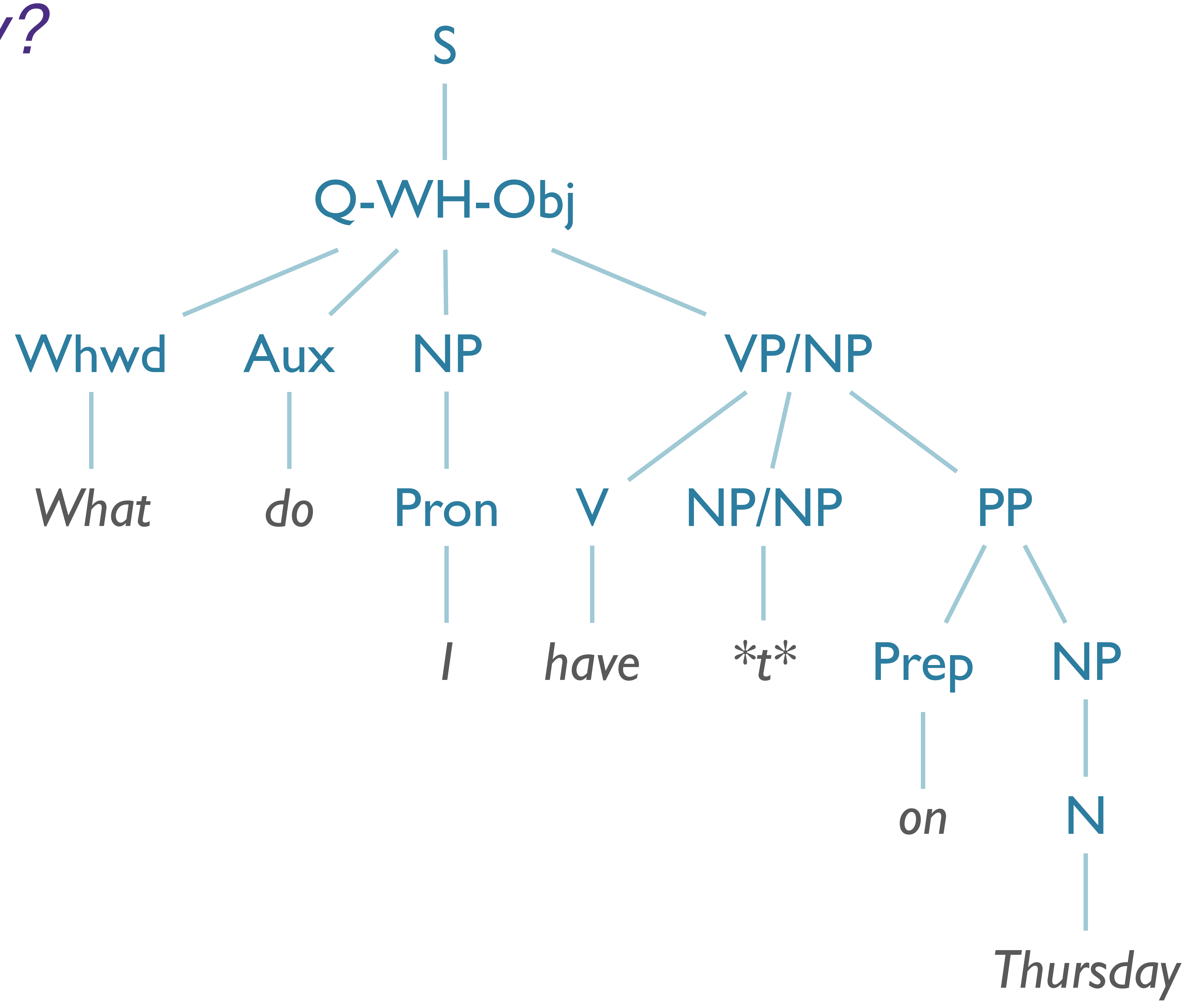
- User: *What do I have on Thursday?*

Dialogue System

- User: *What do I have on Thursday?*
- Parser:
  - Yes! It's grammatical!

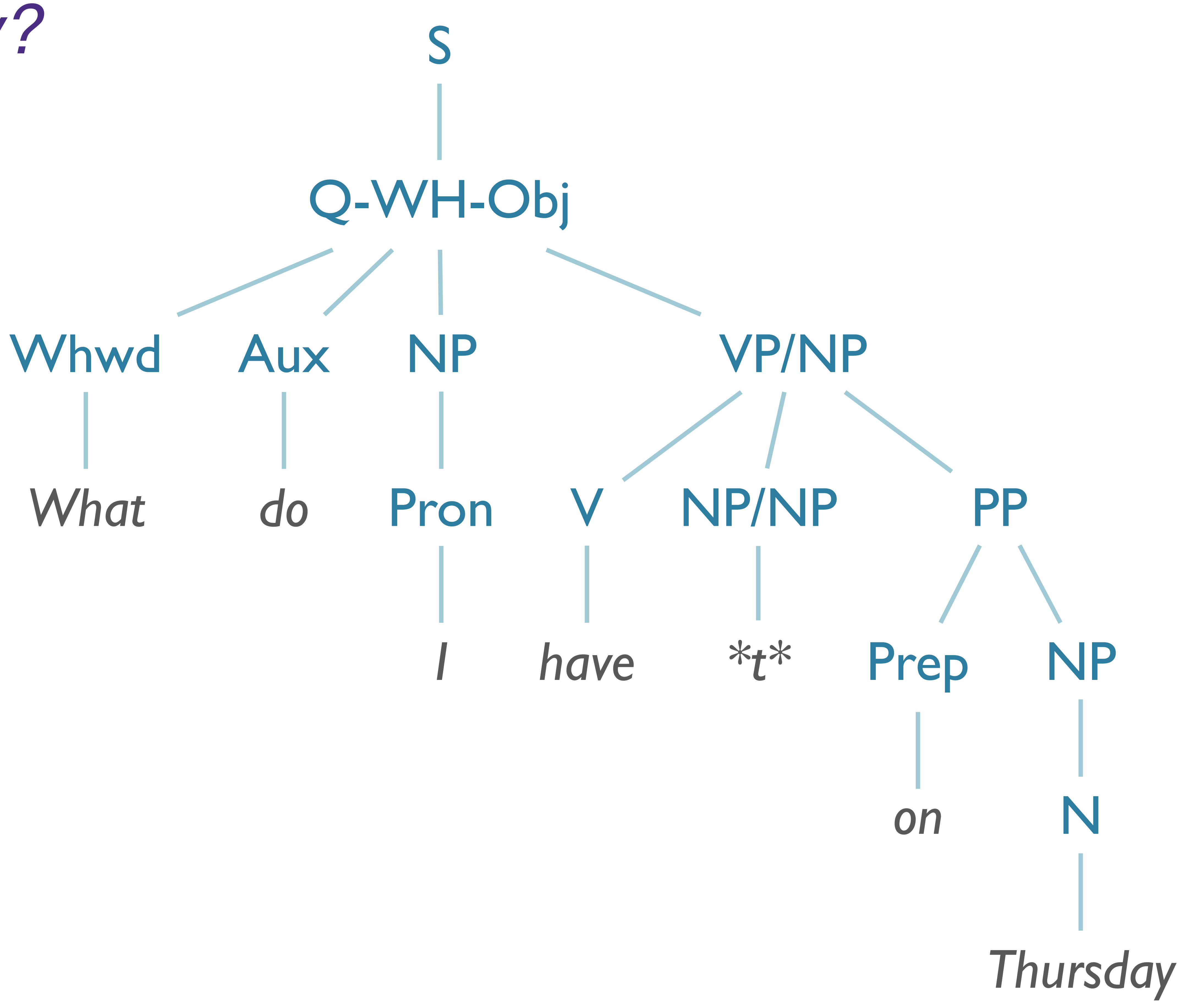
Dialogue System

- User: *What do I have on Thursday?*
- Parser:
  - Yes! It's grammatical!
  - Here's the structure!



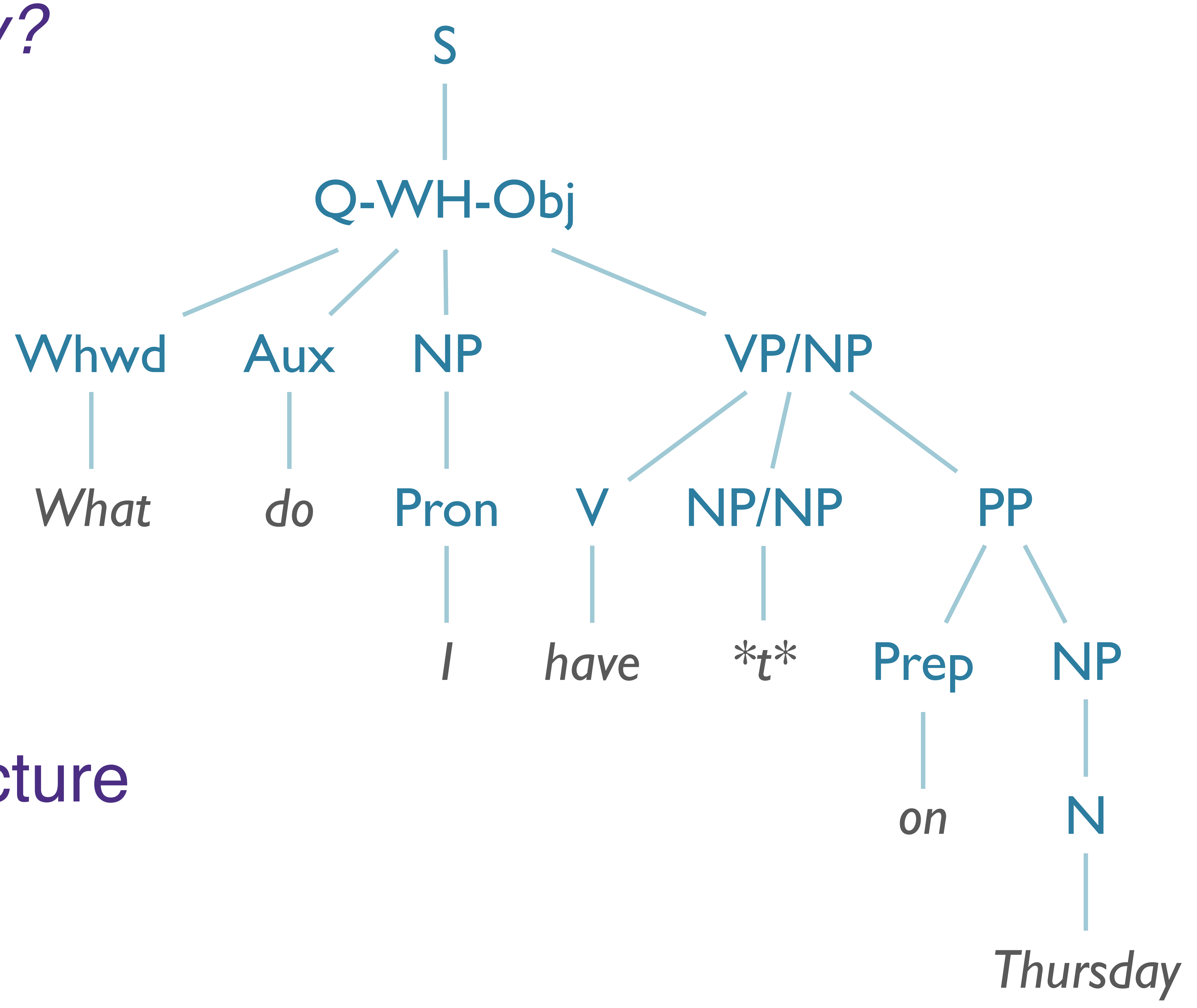
Dialogue System

- User: *What do I have on Thursday?*
- Parser:
  - Yes! It's grammatical!
  - Here's the structure!
- System:
  - Great, but what do I *DO* now?

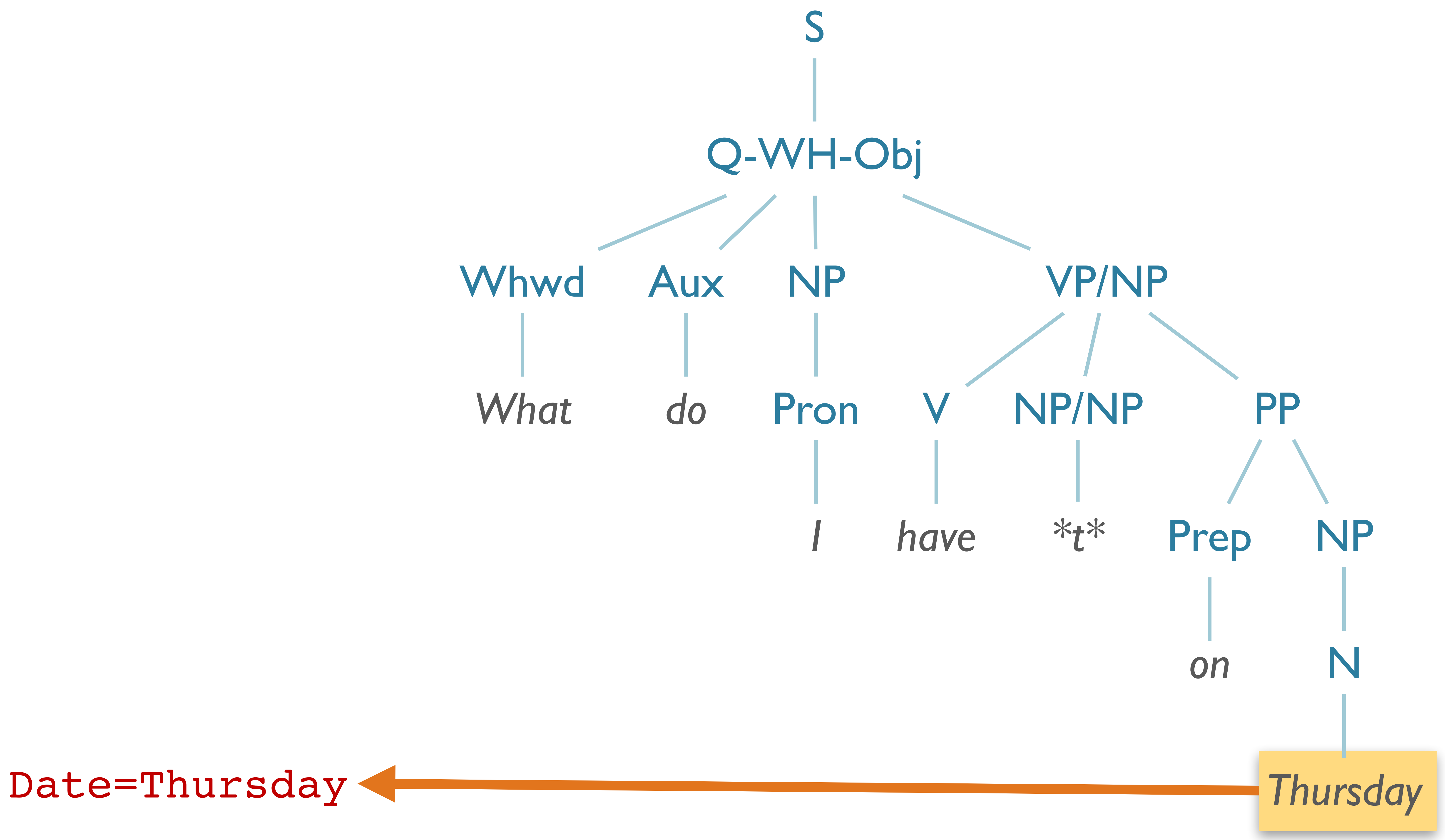


Dialogue System

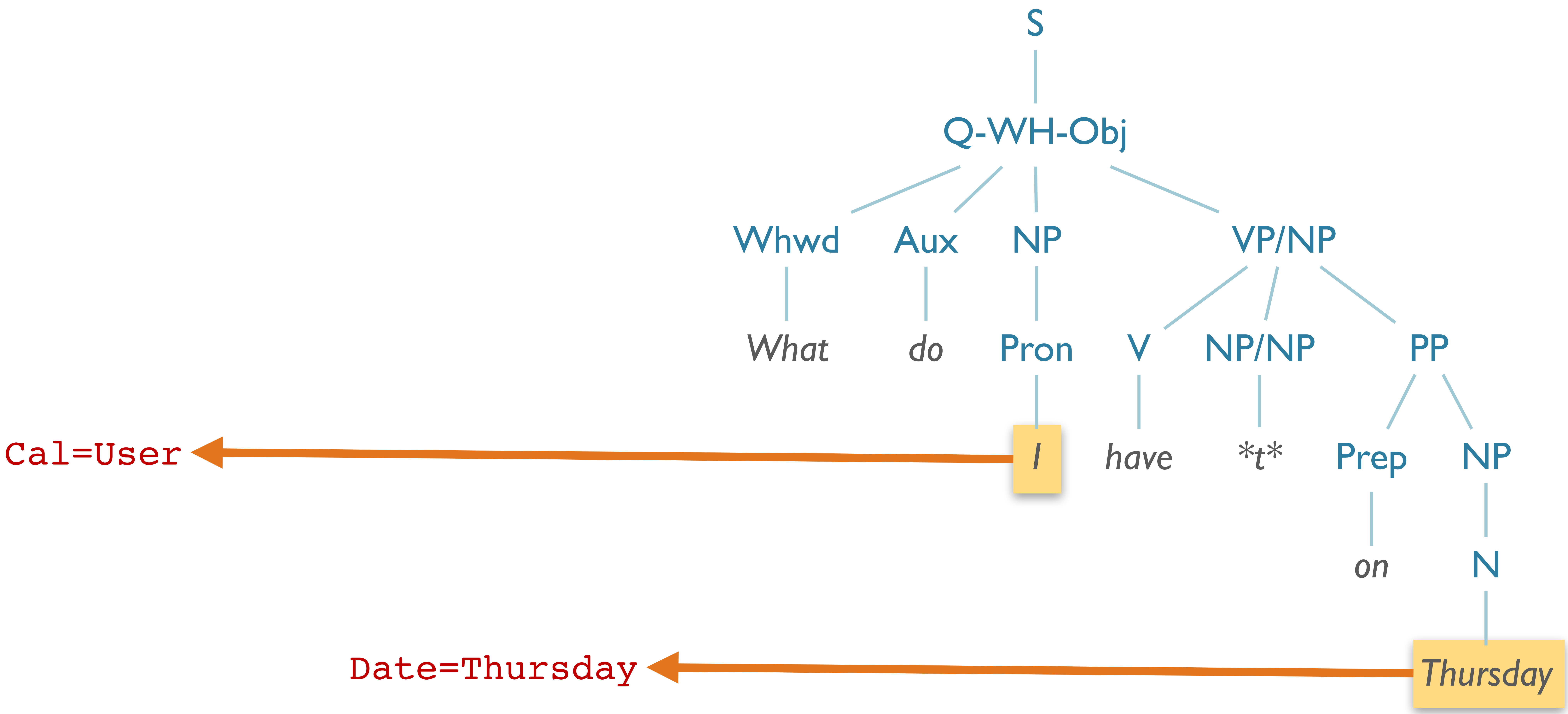
- User: *What do I have on Thursday?*
- Parser:
  - Yes! It's grammatical!
  - Here's the structure!
- System:
  - Great, but what do I *DO* now?
- Need to associate meaning w/structure



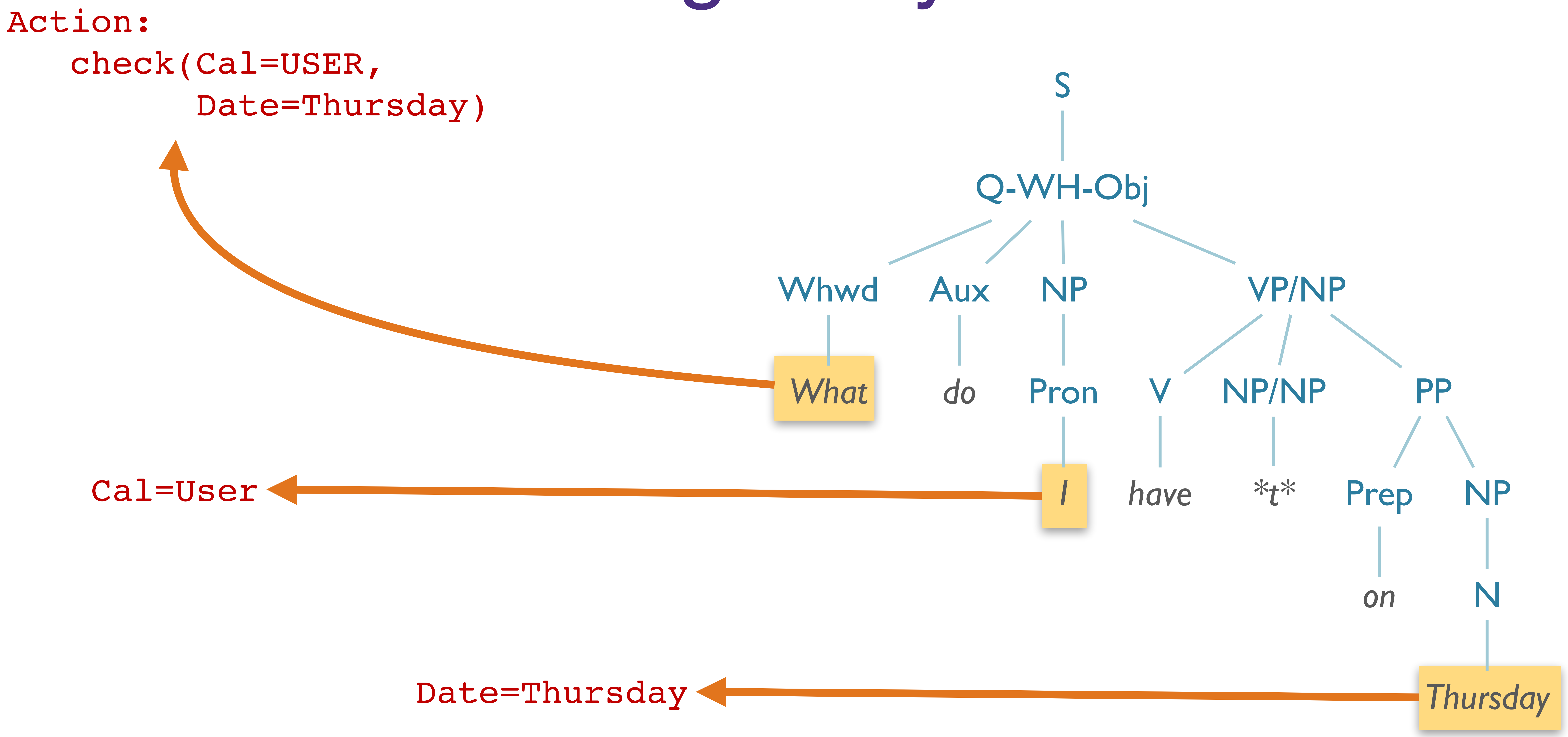
Dialogue System



Dialogue System



Dialogue System



Syntax vs. Semantics

- Syntax:
  - Determine the ***structure*** of natural language input

Syntax vs. Semantics

- Syntax:
  - Determine the ***structure*** of natural language input

- Semantics:
  - Determine the ***meaning*** of natural language input

High-Level Overview

- Semantics = meaning

High-Level Overview

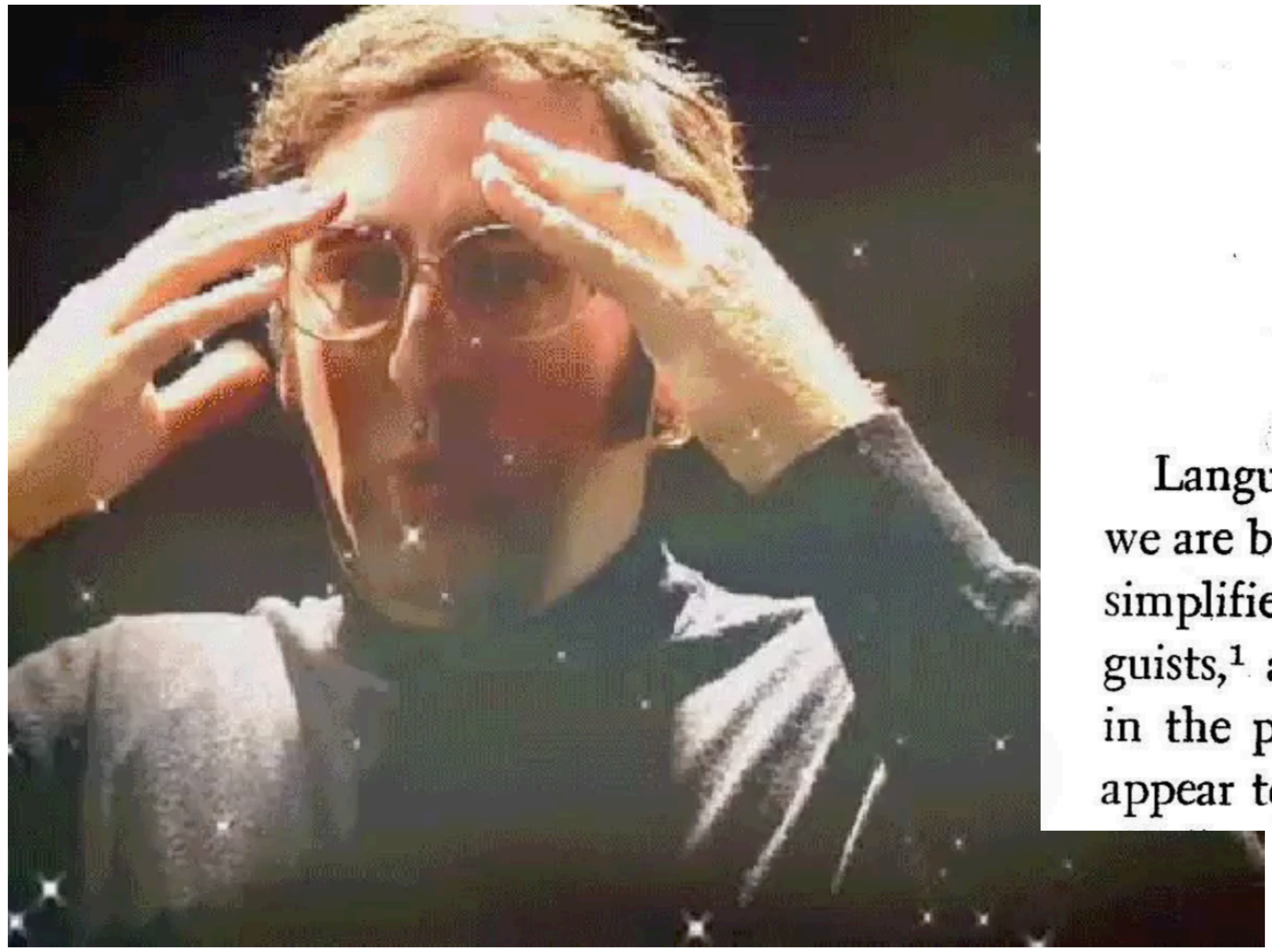
- Semantics = meaning
  - …but what does “meaning” mean?

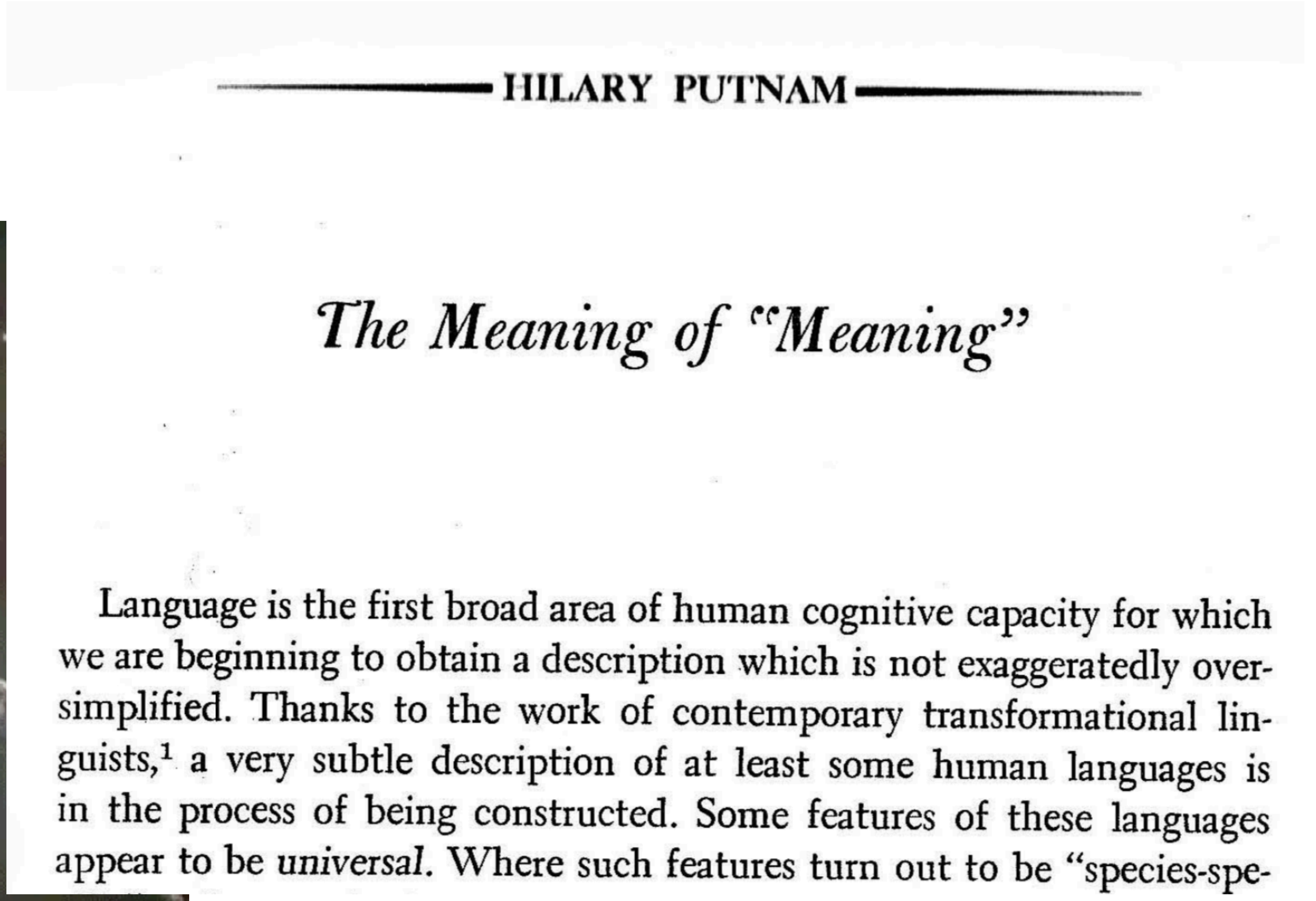
High-Level Overview

- Semantics = meaning
  - …but what does “meaning” mean?

High-Level Overview

- Semantics = meaning
  - …but what does “meaning” mean?





HILARY PUTNAM

The Meaning of “Meaning”

Language is the first broad area of human cognitive capacity for which we are beginning to obtain a description which is not exaggeratedly over-simplified. Thanks to the work of contemporary transformational linguists,[1] a very subtle description of at least some human languages is in the process of being constructed. Some features of these languages appear to be *universal*. Where such features turn out to be “species-spe-

"The sky is blue."

Speech & Text

“The sky is blue.”

Speech & Text

$$\exists x\ Sky(x) \wedge Blue(x)$$

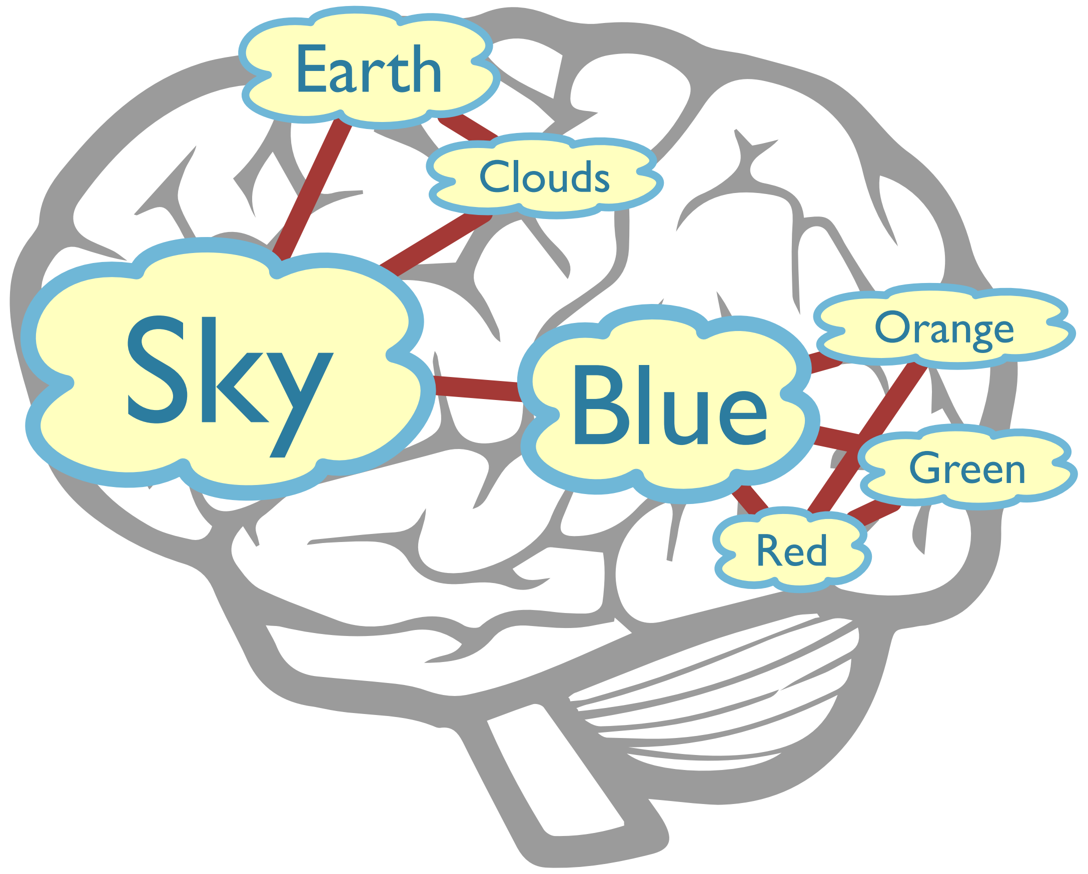
Logic

"The sky is blue."

Speech & Text

$\exists x \; Sky(x) \wedge Blue(x)$

Logic



Psychology

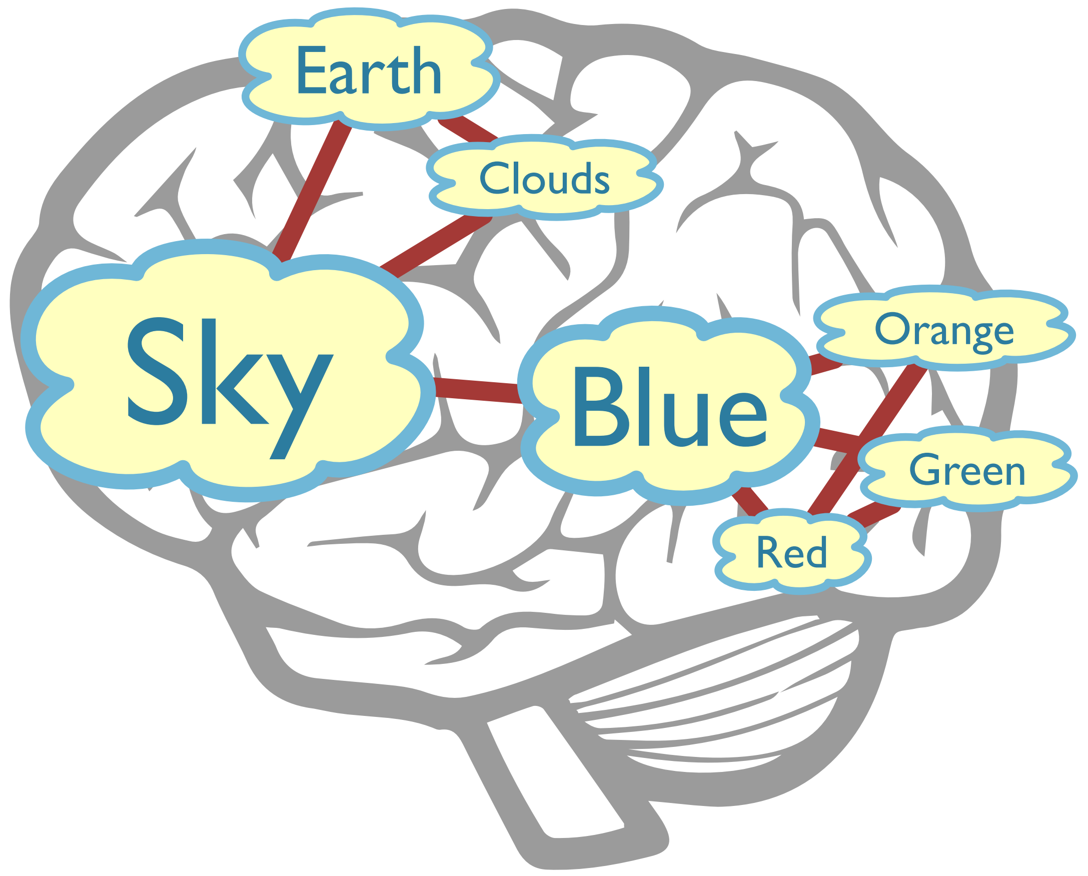
"The sky is blue."

Speech & Text

$$\exists x \; Sky(x) \wedge Blue(x)$$

Logic

Epistemology



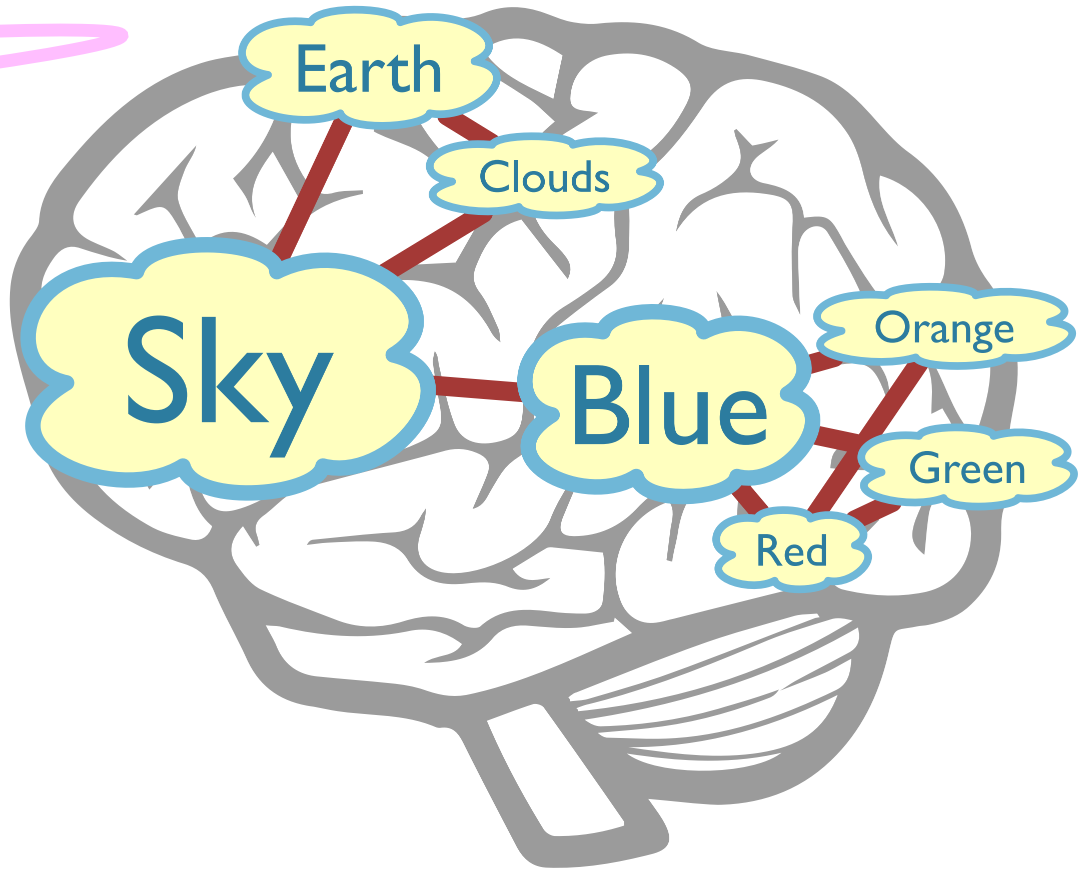
Psychology

“The sky is blue.”

Speech & Text

$$\exists x\ Sky(x) \land Blue(x)$$

Logic



Psychology

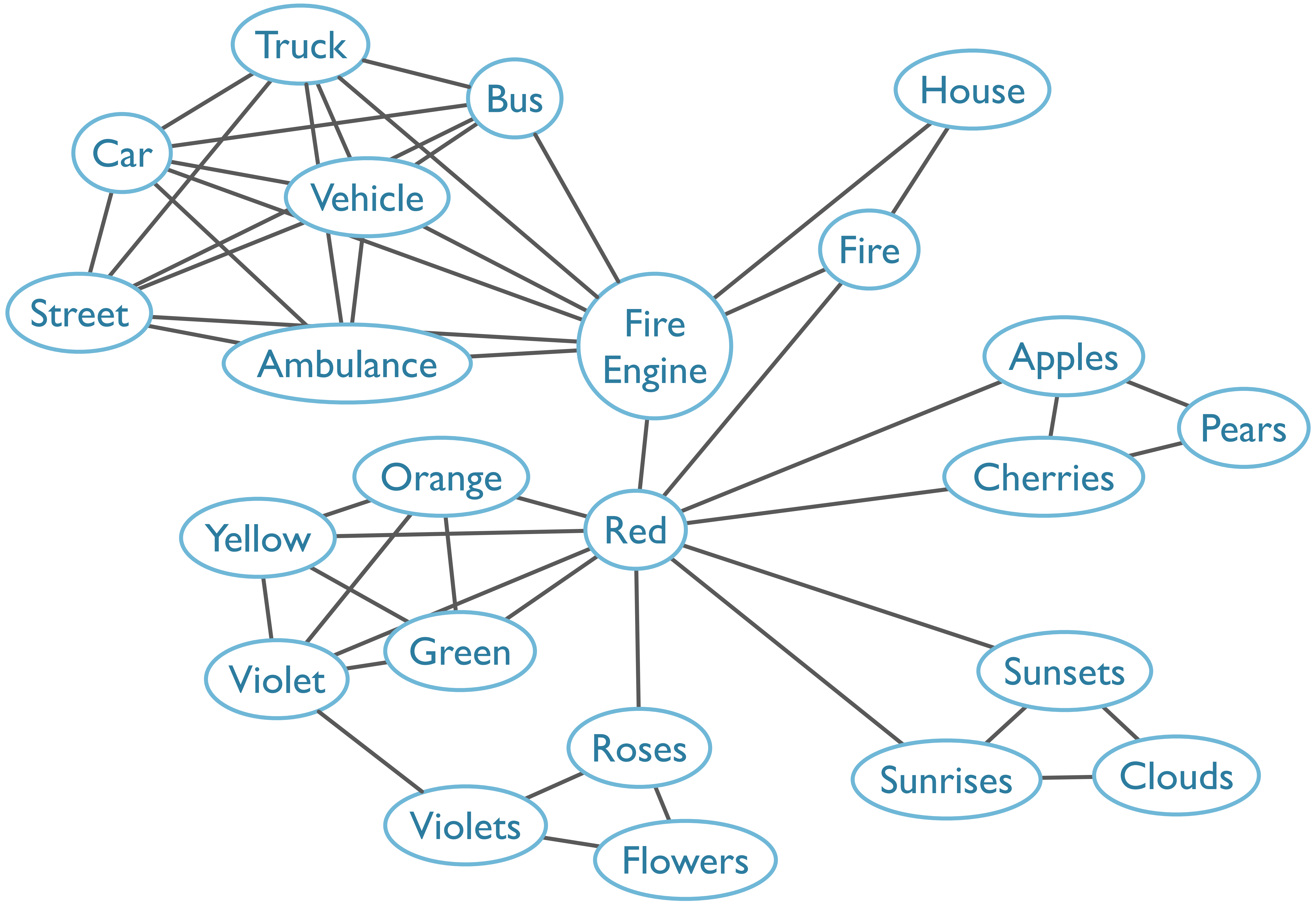
Epistemology

We Will Focus On:

- Concepts that we believe to be true about the world.
- How to connect strings and those concepts.

We ***Won't*** Focus On:

1. Building knowledge bases / semantic networks



Roadmap

- Computational Semantics
  - Overview
  - **Semantics**
  - Representing Meaning
    - First-Order Logic
    - Events
- HW#5
  - Feature grammars in NLTK
  - Practice with animacy

Semantics: an Introduction

Uses for Semantics

- Semantic interpretation required for many tasks
  - Answering questions
  - Following instructions in a software manual
  - Following a recipe
- Requires more than phonology, morphology, syntax
- Must link linguistic elements to world knowledge

Semantics is Complex

- Sentences have many entailments, presuppositions, implicatures
- *Instead, the protests turned bloody, as anti-government crowds were confronted by what appeared to be a coordinated group of Mubarak supporters.*

Semantics is Complex

- Sentences have many entailments, presuppositions, implicatures
- *Instead, the protests turned bloody, as anti-government crowds were confronted by what appeared to be a coordinated group of Mubarak supporters.*
  - The protests *became* bloody.

Semantics is Complex

- Sentences have many entailments, presuppositions, implicatures
- *Instead, the protests turned bloody, as anti-government crowds were confronted by what appeared to be a coordinated group of Mubarak supporters.*
  - The protests *became* bloody.
  - The protests *had been* peaceful.

Semantics is Complex

- Sentences have many entailments, presuppositions, implicatures
- *Instead, the protests turned bloody, as anti-government crowds were confronted by what appeared to be a coordinated group of Mubarak supporters.*
  - The protests *became* bloody.
  - The protests *had been* peaceful.
  - Crowds oppose the government.

Semantics is Complex

- Sentences have many entailments, presuppositions, implicatures
- *Instead, the protests turned bloody, as anti-government crowds were confronted by what appeared to be a coordinated group of Mubarak supporters.*
  - The protests *became* bloody.
  - The protests *had been* peaceful.
  - Crowds oppose the government.
  - Some support Mubarak.

Semantics is Complex

- Sentences have many entailments, presuppositions, implicatures
- *Instead, the protests turned bloody, as anti-government crowds were confronted by what appeared to be a coordinated group of Mubarak supporters.*
  - The protests *became* bloody.
  - The protests *had been* peaceful.
  - Crowds oppose the government.
  - Some support Mubarak.
  - There was a confrontation between two groups.

Semantics is Complex

- Sentences have many entailments, presuppositions, implicatures
- *Instead, the protests turned bloody, as anti-government crowds were confronted by what appeared to be a coordinated group of Mubarak supporters.*
  - The protests *became* bloody.
  - The protests *had been* peaceful.
  - Crowds oppose the government.
  - Some support Mubarak.
  - There was a confrontation between two groups.
  - Anti-government crowds are not Mubarak supporters

Semantics is Complex

- Sentences have many entailments, presuppositions, implicatures
- *Instead, the protests turned bloody, as anti-government crowds were confronted by what appeared to be a coordinated group of Mubarak supporters.*
  - The protests *became* bloody.
  - The protests *had been* peaceful.
  - Crowds oppose the government.
  - Some support Mubarak.
  - There was a confrontation between two groups.
  - Anti-government crowds are not Mubarak supporters
  - …etc.

Challenges in Semantics

- **Semantic Representation**:
  - What is the appropriate formal language to express propositions in linguistic input?
  - e.g.: predicate calculus: $\exists x \left(dog(x) \wedge disappear(x) \right)$

Challenges in Semantics

- **Semantic Representation**:
  - What is the appropriate formal language to express propositions in linguistic input?
  - e.g.: predicate calculus: $\exists x \left(dog\,(x) \wedge disappear\,(x) \right)$
- **Entailment**:
  - What are all the conclusions that can be validly drawn from a sentence?
    - *Lincoln was assassinated* $\vDash$ *Lincoln is dead*
    - $\vDash$ "semantically entails": if former is true, the latter must be too

Challenges in Semantics

- **Reference**
  - How do linguistic expressions link to objects/concepts in the real world?
    - ‘the dog,’ ‘the evening star,’ ‘The Superbowl’

Challenges in Semantics

- **Reference**
  - How do linguistic expressions link to objects/concepts in the real world?
    - ‘the dog,’ ‘the evening star,’ ‘The Superbowl’
- **Compositionality**
  - How can we derive the meaning of a unit from its parts?
  - How do syntactic structure and semantic composition relate?
  - ‘rubber duck’ vs. ‘rubber chicken’ vs. ‘rubber-neck’
  - *kick the bucket*

Tasks in Computational Semantics

- ***Extract***, ***interpret***, and ***reason*** about utterances.

Tasks in Computational Semantics

- ***Extract***, ***interpret***, and ***reason*** about utterances.
- Define a **meaning representation**

Tasks in Computational Semantics

- ***Extract***, ***interpret***, and ***reason*** about utterances.
- Define a **meaning representation**
- Develop techniques for **semantic analysis**
  - …convert strings from natural language to meaning representations

Tasks in Computational Semantics

- ***Extract***, ***interpret***, and ***reason*** about utterances.

- Define a **meaning representation**
- Develop techniques for **semantic analysis**
  - …convert strings from natural language to meaning representations
- Develop methods for **reasoning** about these representations
  - …and performing inference

Tasks in Computational Semantics

- Semantic similarity (words, texts)
- Semantic role labeling
- Semantic analysis / semantic “parsing”
- Recognizing textual entailment (RTE) / natural language inference (NLI)
- Sentiment analysis

Complexity of Computational Semantics

- Knowledge of **language**
  - words, syntax, relationships between structure & meaning, composition procedures

Complexity of Computational Semantics

- Knowledge of **language**
  - words, syntax, relationships between structure & meaning, composition procedures
- Knowledge of **the world**:
  - what are the objects that we refer to?
  - How do they relate?
  - What are their properties?

Complexity of Computational Semantics

- Knowledge of **language**
  - words, syntax, relationships between structure & meaning, composition procedures
- Knowledge of **the world**:
  - what are the objects that we refer to?
  - How do they relate?
  - What are their properties?
- **Reasoning**
  - Given a representation and world, what new conclusions (bits of meaning) can we infer?

Complexity of Computational Semantics

- Effectively AI-complete
  - Needs representation, reasoning, world model, etc.

Representing Meaning

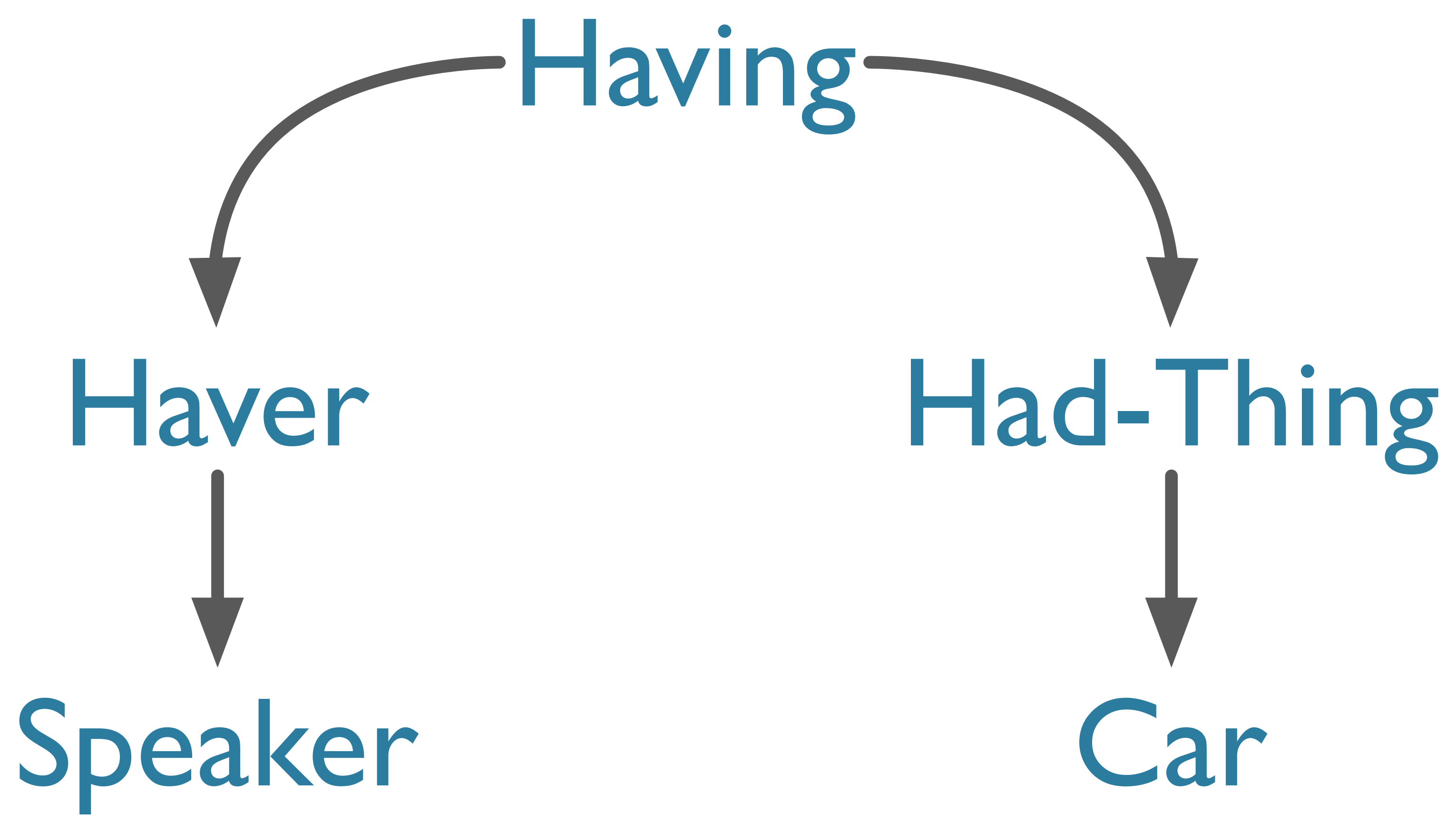
“I have a car”

First-Order Logic: $\exists e, y \left(Having\left(e\right) \wedge Haver\left(e, Speaker\right) \wedge HadThing\left(e, y\right) \wedge Car\left(y\right) \right)$

“I have a car”

First-Order Logic: $\exists e, y\left(Having\left(e\right) \wedge Haver\left(e, Speaker\right) \wedge HadThing\left(e, y\right) \wedge Car\left(y\right)\right)$

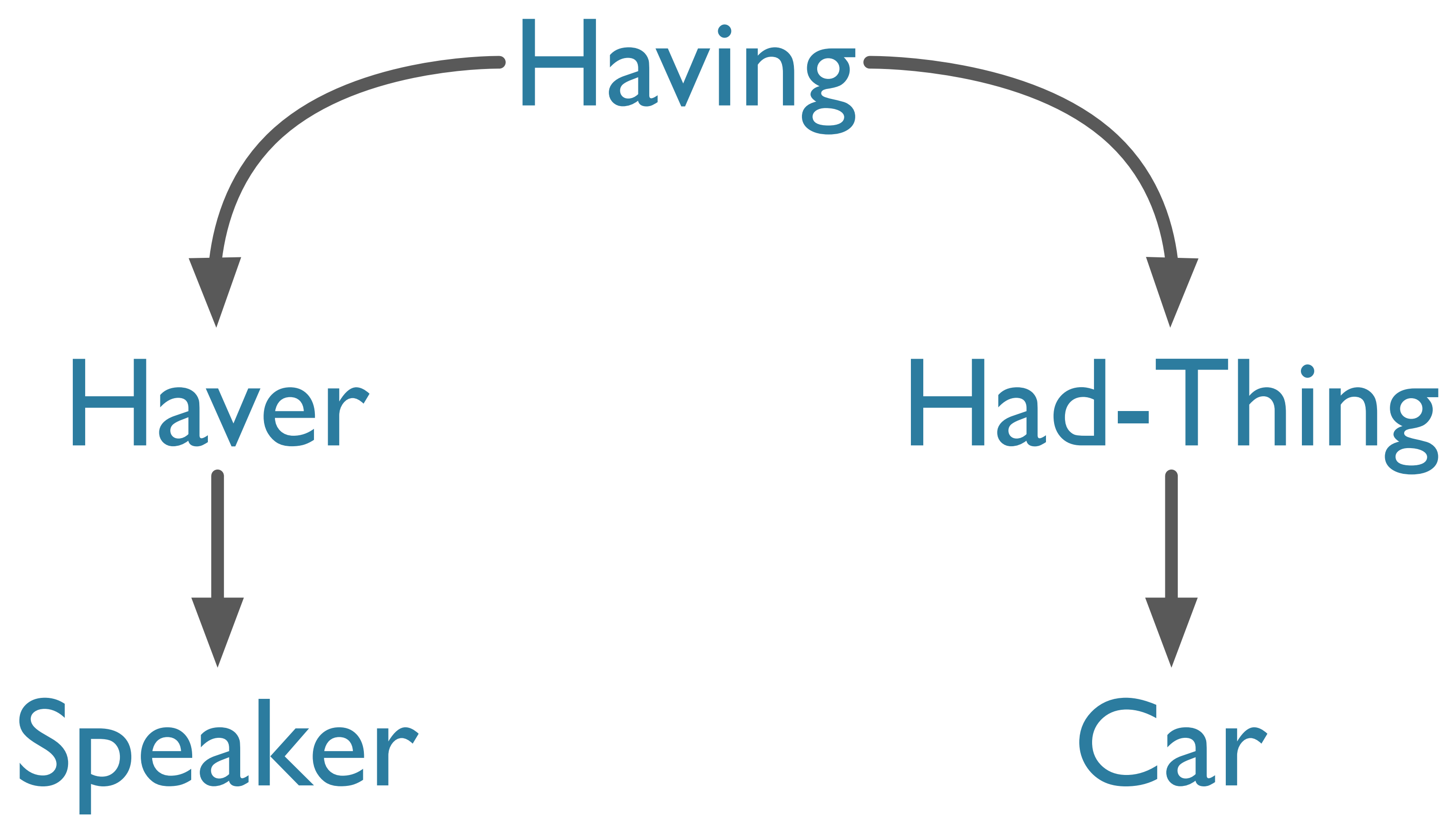
Semantic Network:



"I have a car"

First-Order Logic: $\exists e, y \left(Having\,(e) \wedge Haver\,(e, Speaker) \wedge HadThing\,(e, y) \wedge Car\,(y) \right)$

Semantic Network:



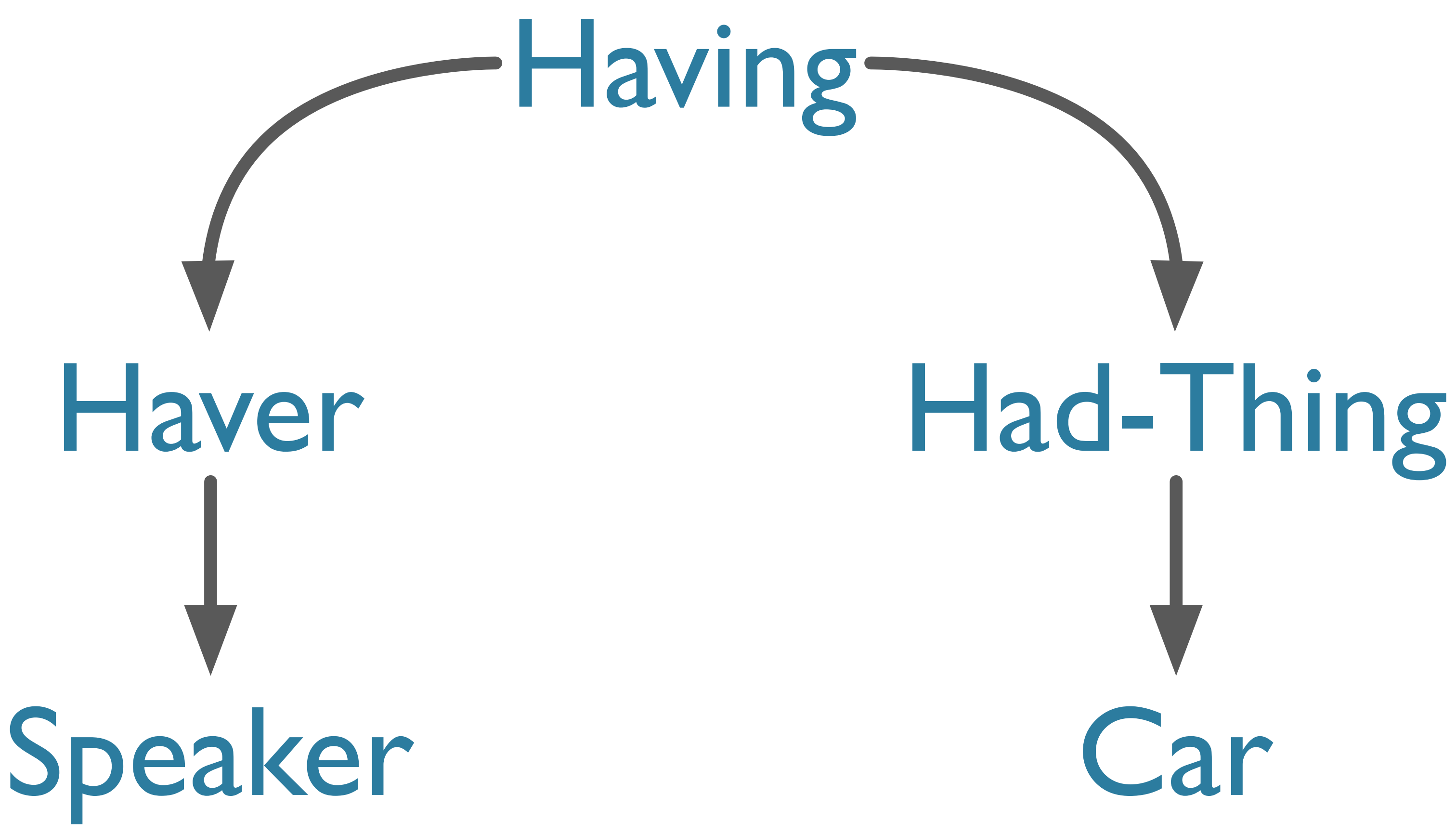
Conceptual Dependency:

$$\begin{array}{l} \textit{Car} \\ \Uparrow \ \textsc{Poss-By} \\ \textit{Speaker} \end{array}$$

"I have a car"

First-Order Logic: $\exists e, y \left(Having\,(e) \wedge Haver\,(e, Speaker) \wedge HadThing\,(e, y) \wedge Car\,(y) \right)$

Semantic Network:



Conceptual Dependency:

$$\begin{array}{c} \textit{Car} \\ \Uparrow \textsc{Poss-By} \\ \textit{Speaker} \end{array}$$

Frame-Based:

```
Having
    Haver: Speaker
    HadThing: Car
```

Meaning Representations

- All consist of structures from set of symbols
  - Representational vocabulary

Meaning Representations

- All consist of structures from set of symbols
  - Representational vocabulary
- Symbol structures correspond to:
  - Objects
  - Properties of objects
  - Relations among objects

Meaning Representations

- All consist of structures from set of symbols
  - Representational vocabulary
- Symbol structures correspond to:
  - Objects
  - Properties of objects
  - Relations among objects
- Can be viewed as:
  - Representation of meaning of linguistic input
  - Representation of state of world

Meaning Representations

- All consist of structures from set of symbols
  - Representational vocabulary
- Symbol structures correspond to:
  - Objects
  - Properties of objects
  - Relations among objects
- Can be viewed as:
  - Representation of meaning of linguistic input
  - Representation of state of world
- Here we focus on **literal** meaning ("what is said")

Representational Requirements

- Verifiability
- Unambiguous representations
- Canonical Form
- Inference and Variables
- Expressiveness

Representational Requirements

- Verifiability
  - Can compare representation of sentence to KB model (generally: "executable")
- Unambiguous representations
- Canonical Form
- Inference and Variables
- Expressiveness

Representational Requirements

- Verifiability
  - Can compare representation of sentence to KB model (generally: "executable")
- Unambiguous representations
  - Semantic representation itself is unambiguous
- Canonical Form
- Inference and Variables
- Expressiveness

Representational Requirements

- Verifiability
  - Can compare representation of sentence to KB model (generally: "executable")
- Unambiguous representations
  - Semantic representation itself is unambiguous
- Canonical Form
  - Alternate expressions of same meaning map to same representation
- Inference and Variables
- Expressiveness

Representational Requirements

- Verifiability
  - Can compare representation of sentence to KB model (generally: "executable")
- Unambiguous representations
  - Semantic representation itself is unambiguous
- Canonical Form
  - Alternate expressions of same meaning map to same representation
- Inference and Variables
  - Way to draw valid conclusions from semantics and KB
- Expressiveness

Representational Requirements

- Verifiability
  - Can compare representation of sentence to KB model (generally: "executable")
- Unambiguous representations
  - Semantic representation itself is unambiguous
- Canonical Form
  - Alternate expressions of same meaning map to same representation
- Inference and Variables
  - Way to draw valid conclusions from semantics and KB
- Expressiveness
  - Represent any natural language utterance

Meaning Structure of Language

- Human Languages:
  - Display basic predicate-argument structure
  - Employ variables
  - Employ quantifiers
  - Exhibit a (partially) compositional semantics

Predicate-Argument Structure

- Represent concepts and relationships

Predicate-Argument Structure

- Represent concepts and relationships
- Some words behave like predicates
  - $\boldsymbol{Book}(John,\ United)$; $\boldsymbol{Non\text{-}stop}(Flight)$

Predicate-Argument Structure

- Represent concepts and relationships
- Some words behave like predicates
  - $\boldsymbol{Book}(John,\ United);\ \boldsymbol{Non\text{-}stop}(Flight)$
- Some words behave like arguments
  - $Book(\boldsymbol{John},\ \boldsymbol{United});\ Non\text{-}stop(\boldsymbol{Flight})$

Predicate-Argument Structure

- Represent concepts and relationships
- Some words behave like predicates
  - $\boldsymbol{Book}(John,\ United);\ \boldsymbol{Non\text{-}stop}(Flight)$
- Some words behave like arguments
  - $Book(\boldsymbol{John},\ \boldsymbol{United});\ Non\text{-}stop(\boldsymbol{Flight})$
- Subcategorization frames indicate:
  - Number, Syntactic category, order of args, possibly other features of args

First-Order Logic

First-Order Logic

- Meaning representation:
  - Provides sound computational basis for verifiability, inference, expressiveness

First-Order Logic

- Meaning representation:
  - Provides sound computational basis for verifiability, inference, expressiveness
- Supports determination of propositional truth

First-Order Logic

- Meaning representation:
  - Provides sound computational basis for verifiability, inference, expressiveness
- Supports determination of propositional truth
- Supports compositionality of meaning*

First-Order Logic

- Meaning representation:
  - Provides sound computational basis for verifiability, inference, expressiveness
- Supports determination of propositional truth
- Supports compositionality of meaning*
- Supports inference

First-Order Logic

- Meaning representation:
  - Provides sound computational basis for verifiability, inference, expressiveness
- Supports determination of propositional truth
- Supports compositionality of meaning*
- Supports inference
- Supports generalization through variables

First-Order Logic Terms

- **Constants**: specific objects in world;
  - $A,\ B,\ John$
  - Refer to exactly one object
  - Each object can have multiple constants refer to it
    - $WAStateGovernor$ and $JayInslee$

First-Order Logic Terms

- **Constants**: specific objects in world;
  - $A,\ B,\ John$
  - Refer to exactly one object
  - Each object can have multiple constants refer to it
    - $WAStateGovernor$ and $JayInslee$
- **Functions**: concepts relating ***objects*** → ***objects***
  - $GovernerOf(WA)$
  - Refer to objects, avoid using constants

First-Order Logic Terms

- **Constants**: specific objects in world;
  - $A,\ B,\ John$
  - Refer to exactly one object
  - Each object can have multiple constants refer to it
    - $WAStateGovernor$ and $JayInslee$
- **Functions**: concepts relating ***objects*** → ***objects***
  - $GovernerOf(WA)$
  - Refer to objects, avoid using constants
- **Variables**:
  - $\boldsymbol{x},\ \boldsymbol{e}$
  - Refer to any potential object in the world

First-Order Logic Language

- **Predicates**
  - Relate ***objects*** to other ***objects***
  - *'United serves Chicago'*
    - $Serves(United,\ Chicago)$

First-Order Logic Language

- **Predicates**
  - Relate ***objects*** to other ***objects***
  - *‘United serves Chicago’*
    - $Serves(United, Chicago)$
- **Logical Connectives**
  - $\{\wedge, \vee, \Rightarrow\}$ = {and, or, implies}
  - Allow for compositionality of meaning* [* many subtleties]
  - *‘Frontier serves Seattle and is cheap.’*
    - $Serves(Frontier, Seattle) \wedge Cheap(Frontier)$

Quantifiers

- $\exists$: **existential** quantifier: *“there exists”*

Quantifiers

- ∃: **existential** quantifier: *"there exists"*
- Indefinite NP
  - ≥**one** such object required for truth

Quantifiers

- ∃: **existential** quantifier: *"there exists"*
- Indefinite NP
  - ≥**one** such object required for truth
- **A non-stop flight** that **serves Pittsburgh**:

$$\exists x\ Flight(x) \land Serves(x, Pittsburgh) \land Non\text{-}stop(x)$$

Quantifiers

- ∀: **universal** quantifier: *"for all"*
  - **All flights include beverages.**

Quantifiers

- $\forall$: **universal** quantifier: *"for all"*
  - **All flights include beverages.**

    $\forall x \; Flight(x) \Rightarrow Includes(x, \; beverages)$

FOL Syntax Summary

$$
\begin{array}{lcl}
\textbf{\textit{Formula}} & \rightarrow & \textit{AtomicFormula} \\
 & | & \textit{Formula Connective Formula} \\
 & | & \textit{Quantifier Variable}, \ldots \textit{Formula} \\
 & | & \neg \textit{Formula} \\
 & | & (\textit{Formula}) \\
\textbf{\textit{AtomicFormula}} & \rightarrow & \textit{Predicate}(\textit{Term}, \ldots) \\
\textbf{\textit{Term}} & \rightarrow & \textit{Function}(\textit{Term}, \ldots) \\
 & | & \textit{Constant} \\
 & | & \textit{Variable}
\end{array}
$$

$$
\begin{array}{lcl}
\textbf{\textit{Connective}} & \rightarrow & \wedge \mid \vee \mid \Rightarrow \\
\textbf{\textit{Quantifier}} & \rightarrow & \forall \mid \exists \\
\textbf{\textit{Constant}} & \rightarrow & \textit{VegetarianFood} \mid \textit{Maharani} \mid \ldots \\
\textbf{\textit{Variable}} & \rightarrow & x \mid y \mid \ldots \\
\textbf{\textit{Predicate}} & \rightarrow & \textit{Serves} \mid \textit{Near} \mid \ldots \\
\textbf{\textit{Function}} & \rightarrow & \textit{LocationOf} \mid \textit{CuisineOf} \mid \ldots
\end{array}
$$

J&M p. 556 (3rd ed. 16.3)

Compositionality

- The meaning of a complex expression is a function of the meaning of its parts, and the rules for their combination.

Compositionality

- The meaning of a complex expression is a function of the meaning of its parts, and the rules for their combination.
- Formal languages **are** compositional.

Compositionality

- The meaning of a complex expression is a function of the meaning of its parts, and the rules for their combination.
- Formal languages **are** compositional.
- Natural language meaning is ***largely compositional,*** though not fully.

Compositionality

- …how can we derive:
  - $loves(John,\ Mary)$

Compositionality

- …how can we derive:
  - $loves(John,\ Mary)$
- from:
  - $John$
  - $loves(x,\ y)$
  - $Mary$

Compositionality

- …how can we derive:
  - $loves(John,\ Mary)$
- from:
  - $John$
  - $loves(x,\ y)$
  - $Mary$
- Lambda expressions!

Lambda Expressions

- Lambda (λ) notation (Church, 1940)
  - Just like lambda in Python, Scheme, etc
  - Allows abstraction over FOL formulae
  - Supports compositionality

- Form: (λ) + variable + FOL expression
  - $\lambda x.P(x)$ "Function taking x to $P(x)$"
  - $\lambda x.P(x)(A) = P(A)$ [called beta-reduction]

λ-Reduction

- λ-reduction: Apply λ-expression to logical term
  - Binds formal parameter to term

$$\lambda x.P(x)$$

λ-Reduction

- λ-reduction: Apply λ-expression to logical term
  - Binds formal parameter to term

$$\lambda x.P(x)$$
$$\lambda x.P(x)(A)$$

λ-Reduction

- λ-reduction: Apply λ-expression to logical term
  - Binds formal parameter to term

$$\lambda x.P(x)$$
$$\lambda x.P(x)(A)$$
$$P(A)$$

λ-Reduction

- λ-reduction: Apply λ-expression to logical term
  - Binds formal parameter to term

$$\lambda x.P(x)$$
$$\lambda x.P(x)(A)$$
$$P(A)$$

- Equivalent to function application

Nested λ-Reduction

- Lambda expression as body of another

$$\lambda x.\lambda y.Near(x,\ y)$$

Nested λ-Reduction

- Lambda expression as body of another

$$\lambda x.\lambda y.Near(x,\ y)$$

$$\lambda x.\lambda y.Near(x,\ y)(Midway)$$

Nested λ-Reduction

- Lambda expression as body of another

$$\lambda x.\lambda y.Near(x,\ y)$$
$$\lambda x.\lambda y.Near(x,\ y)(Midway)$$

Nested λ-Reduction

- Lambda expression as body of another

$$\lambda x.\lambda y.Near(x,\ y)$$

$$\lambda x.\lambda y.Near(x,\ y)(Midway)$$

Nested λ-Reduction

- Lambda expression as body of another

$$\lambda x.\lambda y.Near(x,\ y)$$
$$\lambda x.\lambda y.Near(x,\ y)(Midway)$$
$$\lambda y.Near(Midway,\ y)$$

Nested λ-Reduction

- Lambda expression as body of another

$$\lambda x.\lambda y.Near(x,\ y)$$
$$\lambda x.\lambda y.Near(x,\ y)(Midway)$$
$$\lambda y.Near(Midway,\ y)$$
$$\lambda y.Near(Midway,\ y)(Chicago)$$

Nested λ-Reduction

- Lambda expression as body of another

$$\lambda x.\lambda y.Near(x,\ y)$$
$$\lambda x.\lambda y.Near(x,\ y)(Midway)$$
$$\lambda y.Near(Midway,\ y)$$
$$\lambda y.Near(Midway,\ y)(Chicago)$$

Nested λ-Reduction

- Lambda expression as body of another

$$\lambda x.\lambda y.Near(x,\ y)$$

$$\lambda x.\lambda y.Near(x,\ y)(Midway)$$

$$\lambda y.Near(Midway,\ y)$$

$$\lambda y.Near(Midway,\ y)(Chicago)$$

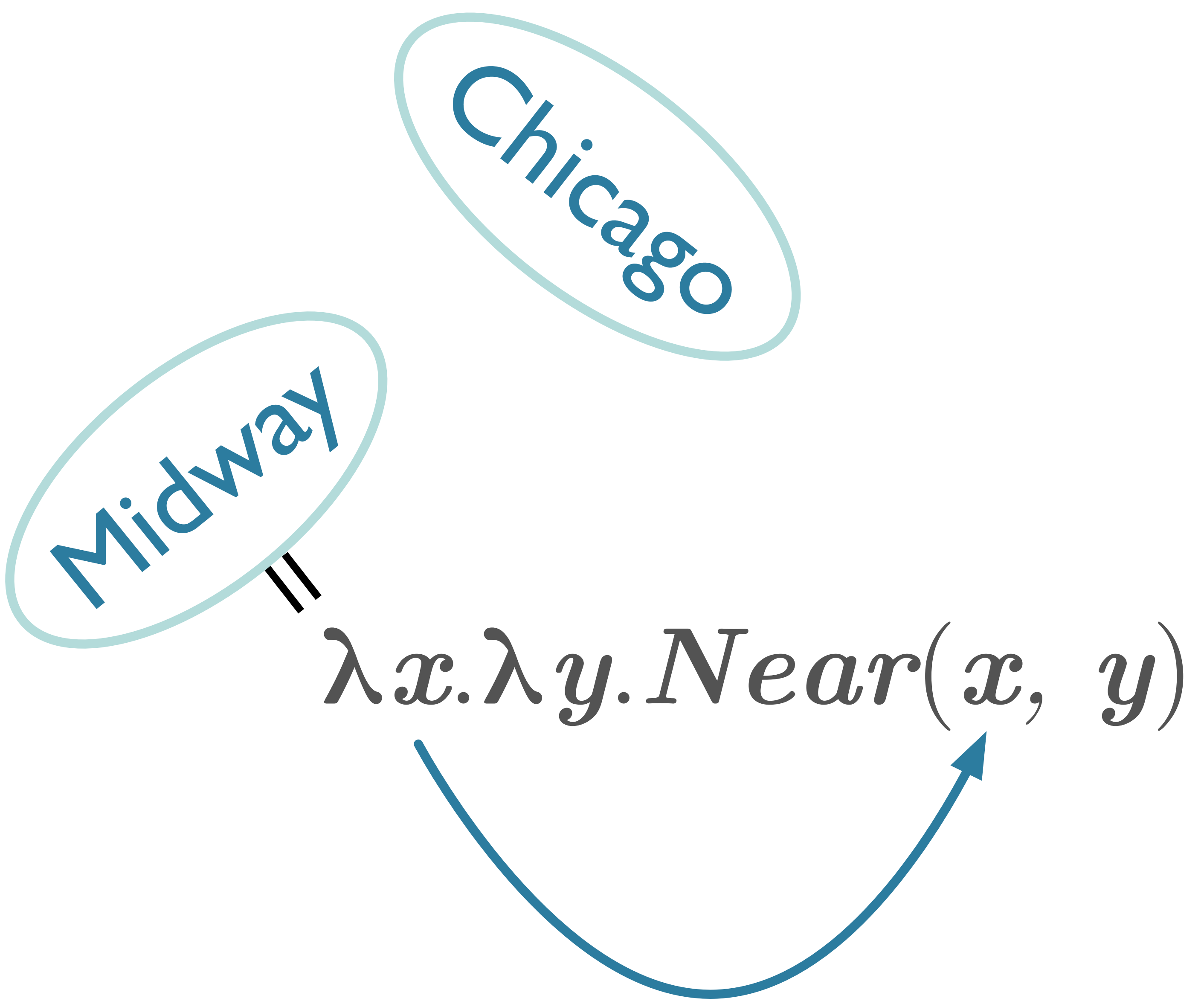
Nested λ-Reduction

- Lambda expression as body of another

$$\lambda x.\lambda y.Near(x,\ y)$$
$$\lambda x.\lambda y.Near(x,\ y)(Midway)$$
$$\lambda y.Near(Midway,\ y)$$
$$\lambda y.Near(Midway,\ y)(Chicago)$$
$$Near(Midway,\ Chicago)$$

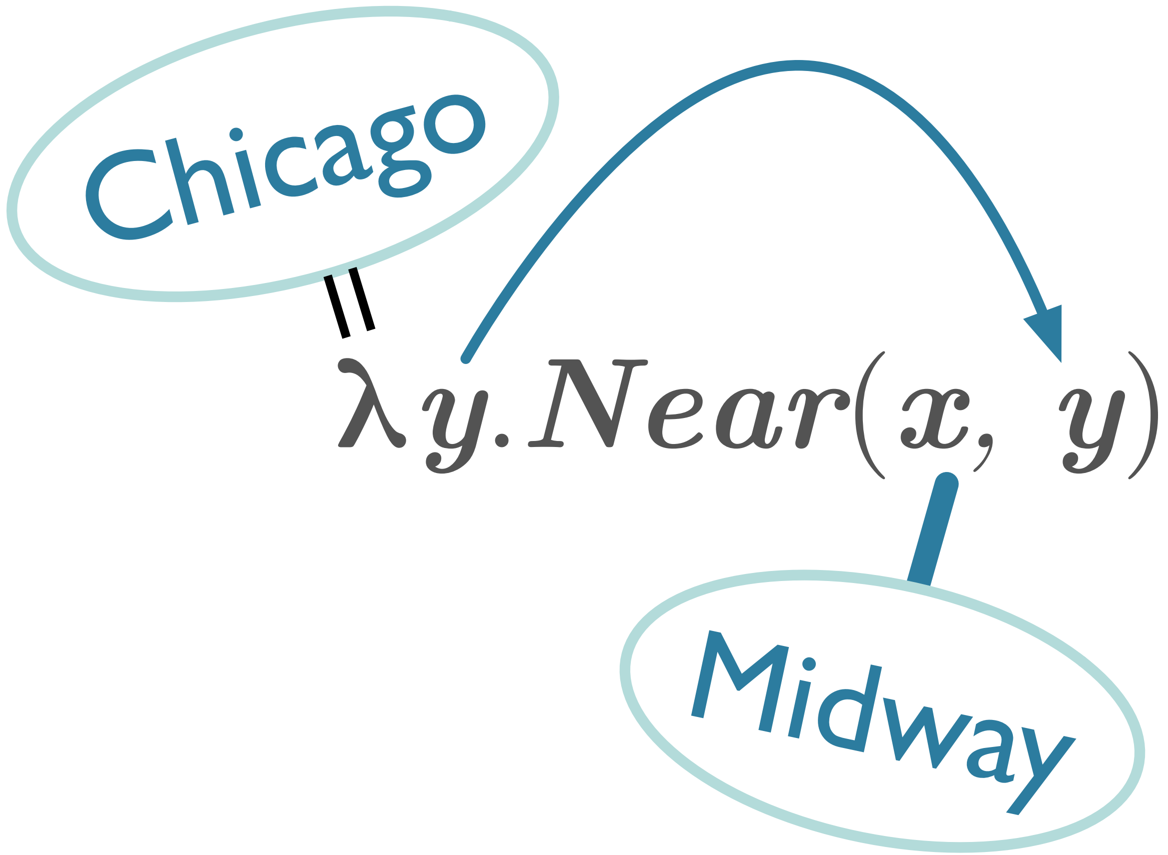
Nested λ-Reduction

- If it helps, think of λs as binding sites:



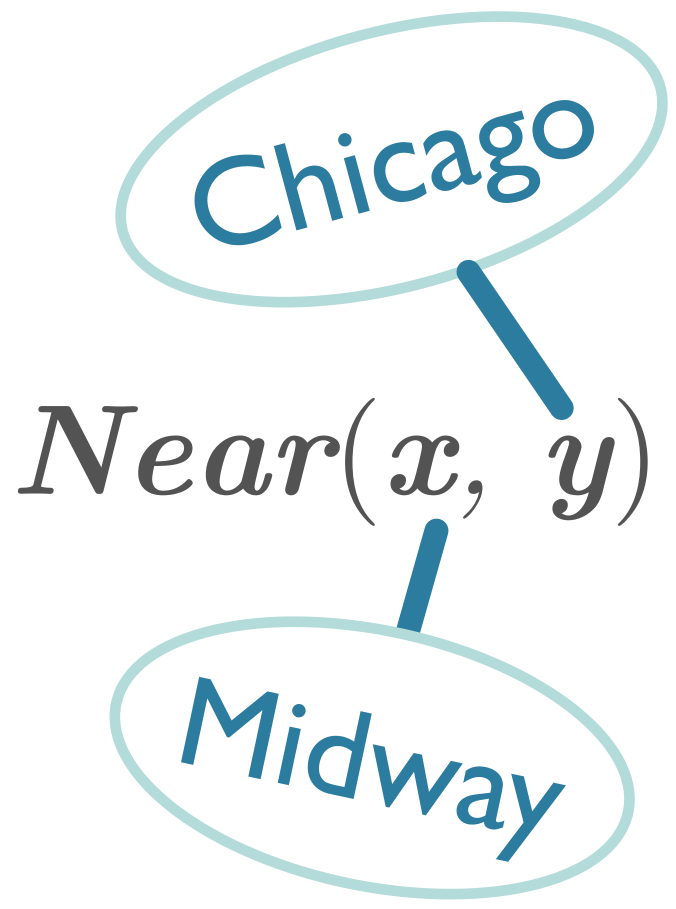
Nested λ-Reduction

- If it helps, think of λs as binding sites:



Nested λ-Reduction

- If it helps, think of λs as binding sites:



Lambda Expressions

- ***Currying***
  - Converting multi-argument predicates to sequence of single argument predicates
  - Why?
    - Incrementally accumulates multiple arguments spread over different parts of parse tree

Lambda Expressions

- ***Currying***
  - Converting multi-argument predicates to sequence of single argument predicates
  - Why?
    - Incrementally accumulates multiple arguments spread over different parts of parse tree

- …or *Schönkfinkelization*

Logical Formulae

- FOL terms (objects): denote elements in a domain
  - Properties: sets of domain elements
  - Relations: sets of tuples of domain elements

Logical Formulae

- FOL terms (objects): denote elements in a domain
  - Properties: sets of domain elements
  - Relations: sets of tuples of domain elements
- Atomic formulae: P(x), R(x,y), etc

Logical Formulae

- FOL terms (objects): denote elements in a domain
  - Properties: sets of domain elements
  - Relations: sets of tuples of domain elements
- Atomic formulae: P(x), R(x,y), etc
- Formulae based on logical operators:

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$
F	F	T	F	F	T
F	T	T	F	T	T
T	F	F	F	T	F
T	T	F	T	T	T

Logical Formulae: Finer Points

- $\vee$ is not exclusive:
  - *Your choice is pepperoni or sausage*
  - …use $\veebar$ or $\oplus$

Logical Formulae: Finer Points

- $\vee$ is not exclusive:
  - *Your choice is pepperoni or sausage*
  - …use $\veebar$ or $\oplus$
- $\Rightarrow$ is the logical form
  - Does not mean the same as natural language "if", just that if LHS=T, then RHS=T

Inference

1. α

1. $\forall x\ \alpha(x)$

Inference

1. α
2. $\alpha \Rightarrow \beta$

1. $\forall x\ \alpha(x)$

Inference

1. α
2. $\alpha \Rightarrow \beta$
3. $\therefore \beta$

1. $\forall x\ \alpha(x)$

Inference

1. α
2. $\alpha \Rightarrow \beta$
3. $\therefore \beta$

1. $\forall x\ \alpha(x)$
2. $\therefore \alpha(t)$

Inference

1. $VegetarianRestaurant(Leaf)$

Inference

1. $VegetarianRestaurant(Leaf)$

2. $\forall x \; VegetarianRestaurant(x) \Rightarrow Serves(x, VegetarianFood)$

Inference

1. $VegetarianRestaurant(Leaf)$

2. $\forall x \;\; VegetarianRestaurant(x) \Rightarrow Serves(x, VegetarianFood)$

3. $\therefore \; Serves(Leaf, \;\; VegetarianFood)$

Inference

- Standard AI-type logical inference procedures
  - Modus Ponens
  - Forward-chaining, Backward Chaining
  - Abduction
  - Resolution
  - Etc…

Inference

- Standard AI-type logical inference procedures
  - Modus Ponens
  - Forward-chaining, Backward Chaining
  - Abduction
  - Resolution
  - Etc…
- We’ll assume we have a theorem prover.

Roadmap

- Computational Semantics
  - Introduction
  - Semantics
  - Representing Meaning
    - First-Order Logic
    - **Events**
- HW#5
  - Feature grammars in NLTK
  - Practice with animacy

Events

Representing Events

- Initially, single predicate with some arguments
  - $Serves(United, Houston)$
  - Assume # of args = # of elements in subcategorization frame

Representing Events

- Initially, single predicate with some arguments
  - $Serves(United,\ Houston)$
  - Assume # of args = # of elements in subcategorization frame
- Example:
  - *The flight arrived*
  - *The flight arrived in Seattle*
  - *The flight arrived in Seattle on Saturday.*
  - *The flight arrived on Saturday.*
  - *The flight arrived in Seattle from SFO.*
  - *The flight arrived in Seattle from SFO on Saturday.*

Representing Events

- Initially, single predicate with some arguments
  - $Serves(United, Houston)$
  - Assume # of args = # of elements in subcategorization frame
- Example:
  - *The flight arrived*
  - *The flight arrived in Seattle*
  - *The flight arrived in Seattle on Saturday.*
  - *The flight arrived on Saturday.*
  - *The flight arrived in Seattle from SFO.*
  - *The flight arrived in Seattle from SFO on Saturday.*
- Variable number of arguments; many entailment relations here.

Representing Events

- ***Arity***:
  - How do we deal with different numbers of arguments?

Representing Events

- ***Arity***:
  - How do we deal with different numbers of arguments?

- *The flight arrived in Seattle from SFO on Saturday.*

Representing Events

- ***Arity***:
  - How do we deal with different numbers of arguments?

- *The flight arrived in Seattle from SFO on Saturday.*
  - Davidsonian (Davidson 1967):
    - $\exists \boldsymbol{e}\ Arrival(\boldsymbol{e}, Flight, Seattle, SFO) \wedge Time(\boldsymbol{e}, Saturday)$

Representing Events

- ***Arity***:
  - How do we deal with different numbers of arguments?

- *The flight arrived in Seattle from SFO on Saturday.*
  - Davidsonian (Davidson 1967):
    - $\exists \boldsymbol{e}\ Arrival(\boldsymbol{e}, Flight, Seattle, SFO) \land Time(\boldsymbol{e}, Saturday)$
  - Neo-Davidsonian (Parsons 1990):
    - $\exists \boldsymbol{e}\ Arrival(\boldsymbol{e}) \land Arrived(\boldsymbol{e}, Flight) \land Destination(\boldsymbol{e}, Seattle) \land Origin(\boldsymbol{e}, SFO) \land Time(\boldsymbol{e}, Saturday)$

Why events?

- “Adverbial modification is thus seen to be logically on a par with adjectival modification: what adverbial clauses modify is not verbs but the events that certain verbs introduce.” —Davidson

Neo-Davidsonian Events

- Neo-Davidsonian representation:
  - Distill event to single argument for event itself
  - Everything else is additional predication

Neo-Davidsonian Events

- Neo-Davidsonian representation:
  - Distill event to single argument for event itself
  - Everything else is additional predication
- Pros
  - No fixed argument structure
  - Dynamically add predicates as necessary
  - No unused roles
  - Logical connections can be derived

Meaning Representation for Computational Semantics

- Requirements
  - Verifiability
  - Unambiguous representation
  - Canonical Form
  - Inference
  - Variables
  - Expressiveness
- Solution:
  - First-Order Logic
    - Structure
    - Semantics
    - Event Representation

Summary

- FOL can be used as a meaning representation language for natural language
- Principle of compositionality:
  - The meaning of a complex expression is a function of the meaning of its parts
- λ-expressions can be used to compute meaning representations from syntactic trees based on the principle of compositionality
- In next classes, we will look at syntax-driven approach to semantic analysis in more detail

HW #4

Probabilistic Parsing

- Goals:
  - Learn about PCFGs
  - Implement PCKY
  - Analyze Parsing Evaluation
  - Assess improvements to PCFG Parsing

Tasks

1. Train a PCFG
    1. Estimate rule probabilities from treebank
    2. Treebank is already in CNF
    3. More ATIS data from Penn Treebank
2. Build CKY Parser
    1. Modify (your) existing CKY implementation

Tasks

3. Evaluation
    1. Evaluate your parser using standard metric
    2. We will provide **evalb** program and gold standard
4. Improvement
    1. Improve your parser in some way:
        1. Coverage
        2. Accuracy
        3. Speed
    2. Evaluate new parser

Improvement Possibilities

- Coverage:
  - Some test sentences won't parse as is!
    - Lexical gaps (aka out-of-vocabulary [OOV] tokens)
      - …remember to model the probabilities, too
- Better context modeling
  - e.g. — Parent Annotation
- Better Efficiency
  - e.g. — Heuristic Filtering, Beam Search
- No "cheating" improvements:
  - improvement can't change training by looking at test data

`evalb`

- `evalb` available in `dropbox/21-22/571/hw4/tools`
- `evalb […] <gold-file> <test-file>`
- `evalb --help` for more info
- NB: specify **full/absolute path** to evalb when invoking in your scripts